The Orthogonal Band Decomposition of the Finite Dirichlet Matrix and its Applications

A Thesis Submitted to the
College of Graduate and Postdoctoral Studies
in Partial Fulfillment of the Requirements
for the degree of Doctor of Philosophy
in the Department of Mathematics and Statistics
University of Saskatchewan
Saskatoon

By
Izabela Vlahu

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Mathematics and Statistics
McLean Hall
106 Wiggins Road
University of Saskatchewan
Saskatoon, Saskatchewan S7N 5E6
Canada

OR

Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9
Canada

Abstract

In my work I establish and extend the theory of finite D-matrices for the purposes of signal processing applications in the finite, digital setting. Finite D-matrices are obtained by truncating infinite D-matrices to upper-left corners. I show that finite D-matrices are furnished with a number-theoretical structure that is not present in their infinite counterparts. In particular, I show that the columns of every finite D-matrix of size $N \times N$ admits a natural, non-trivial, Orthogonal Band Decomposition, induced by the Floor Band Decomposition on the finite set $\{1, 2, \ldots, N\}$. When the D-matrix is invertible, its Orthogonal Band Decomposition induces a non-trivial resolution of the identity. Furthermore, for every finite D-matrix A, I show that the sum P of the orthogonal projections corresponding to each band of A admits the following sparse representation $P = A\Lambda^{-1}A^*$, where Λ is a special diagonal matrix and $A^\star$ is the Hermitian adjoint of A. I also show that the matrix P and its inverse induce another non-trivial resolution of the identity. Being a sum of projection matrices, I call the matrix P the associated P-matrix of A.

Both the finite D-matrices and their associated P-matrices can be applied in the processing of digital signals. For example, given a D-matrix A, its associated P-matrix allows us to pass from a signal representation in the Fourier basis to a representation, as a sum of projections, in the basis induced by the Orthogonal Band Decomposition of A. Preliminary experiments suggest that the error of approximating signals with partial sums of projections might offer a more suitable metric to choose D-matrix representations in specific applications. Significantly, computations with finite D-matrices and P-matrices can be carried out via fast algorithms, which makes these transforms computationally competitive.

Acknowledgements

I would like to express my deepest gratitude to my supervisors Professors Artur Sowa and Paul Babyn for their invaluable guidance and support during the course of my studies and the preparation of this thesis. Without their patience and genuine interest in my work, I would not have been able to write this work.

I would also like to thank all my committee members for taking the time and effort to read my work and provide their own perspectives. I believe that with their comments and suggestions I was able to improve the presentation of this thesis.

Finally, I would like to thank my family for their tremendous support and encouragement during my studies. Without my mother's help and my son's patience and understanding, I would not me able to complete this degree.

To my mother Suzana, my brother Petar and my son Naum

Contents

List of Figures

Chapter 1

Introduction

Signals are functions that represent physical quantities or variables. When signals are represented by functions of continuous variables, we refer to them as analog signals. Alternatively, when signals are represented by functions of discrete variables, which have finitely many values, we refer to them as digital signals. In practice, we can think of digital signals as finite samplings of analog signals. On the computer, we store one dimensional digital signals as vectors and we store two dimensional digital signals as matrices. The applications of this work focus on the finite framework of digital signals.

To study signals mathematically, we represent them as combinations of carefully chosen functions. We refer to these combinations as basis representations. If we are interested in studying signals through their frequency properties, for example, the standard approach is to consider their representations in the Fourier basis, which consists of sinusoidal waves $\{e^{i(nt)} \mid n \in \mathbb{Z}\}$. Recently, in [22], [24] and [27], Sowa showed that it is possible to generalize the Fourier basis to a basis of the form $\{f(nt) \mid n \in \mathbb{Z}\}$, with some moderate restrictions on the generating function f. The importance of Sowa's work is that it allows us to customize bases to particular signals or families of signals. His setting addresses the case of continuous functions. The purpose of this work, however, is to establish and extend the theory in the digital setting.

Passage between representations of functions in the Fourier basis and representations in a basis $\{f(nt)\}$, both in the finite and in the infinite case, is achieved via matrix multiplication by a special matrix called a D-matrix. The form of a D-matrix, in the infinite case, is as follows:

$$A = \begin{bmatrix} \alpha_1 & & & & & & & & \cdots \\ \alpha_2 & \alpha_1 & & & & & & & \cdots \\ \alpha_3 & & \alpha_1 & & & & & & \cdots \\ \alpha_4 & \alpha_2 & & \alpha_1 & & & & & \cdots \\ \alpha_5 & & & & \alpha_1 & & & & \cdots \\ \alpha_6 & \alpha_3 & \alpha_2 & & & \alpha_1 & & & \cdots \\ \alpha_7 & & & & & & \alpha_1 & & \cdots \\ \alpha_8 & \alpha_4 & & \alpha_2 & & & & \alpha_1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{1.0.1}$$

Every column of the matrix A exhibits the same sequence (α_n) of complex numbers, where the m^{th} column has $m-1$ zeros inserted between the sequence terms. One can think of the second column as being obtained

from the first via "stretching" by a factor of 2. Similarly, one can think of the third column as being obtained from the first via "stretching" by a factor of 3. Intuitively, we can see that the first column represents the function $f(t)$, the second column represents the function $f(2t)$, the third $f(3t)$, and so on. To be precise, the sequence (α_n) consists of the Fourier coefficients of the basis function f.

Computations with D-matrices are carried out via finite truncations to upper-left corners of size $N \times N$. These truncations are what we refer to as finite D-matrices. Significantly, computations with finite D-matrices can be carried out via fast algorithms of complexity $\mathcal{O}(N \log N)$. This makes D-matrices computationally competitive.

When we consider D-matrices for signal processing applications of digital signals, it is precisely the finite D-matrices that are of interest. As we will see in this thesis, the finite D-matrices exhibit a richer number-theoretical structure than their infinite counterparts in that the columns of finite D-matrices admit a natural, non-trivial, Orthogonal Band Decomposition. Consider, for example, the following finite D-matrix A of size 8×8:

$$A = \begin{bmatrix} \alpha_1 & & & & & & & \\ \alpha_2 & \alpha_1 & & & & & & \\ \alpha_3 & & \alpha_1 & & & & & \\ \alpha_4 & \alpha_2 & & \alpha_1 & & & & \\ \alpha_5 & & & & \alpha_1 & & & \\ \alpha_6 & \alpha_3 & \alpha_2 & & & \alpha_1 & & \\ \alpha_7 & & & & & & \alpha_1 & \\ \alpha_8 & \alpha_4 & & \alpha_2 & & & & \alpha_1 \end{bmatrix}. \tag{1.0.2}$$

Observe that the third and fourth columns of the matrix A are mutually orthogonal and their non-trivial terms are the first two elements of the sequence (α_n). In fact, if we partition the columns of A according to the number of non-trivial terms, the columns of each partition class will be mutually orthogonal and of equal norm. In Theorem 4.2.2, I show that every finite D-matrix of size $N \times N$ admits a natural partition into orthogonal classes of columns of equal norm. Consequently, I call this partition the Orthogonal Band Decomposition. Furthermore, in Theorem 4.2.2, I show that the Orthogonal Band Decomposition of an invertible D-matrix A induces a non-trivial resolution of the identity of the form

$$I = \sum_k \lambda_k^{-1} A_k^* A_k, \tag{1.0.3}$$

where A_k is the $N \times N$ matrix whose non-zero columns correspond to the columns in the k^{th} band of A. Furthermore, the matrix A_k^* denotes the conjugate transpose of A_k, while λ_k denotes the square of the norm of the columns in the corresponding band.

Note that the Orthogonal Band Decomposition of the finite D-matrix A does not depend on the sequence (α_n). Indeed, the decomposition depends solely on the size of the matrix and this is where the richness of the number-theoretical structure becomes apparent. Consider the finite set $\{1, 2, \ldots, N\}$. Think of it as the set of

indices of the columns of a D-matrix of size $N \times N$. Now consider the floor values $\lfloor N/k \rfloor$, for $k \in \{1, 2, \ldots, N\}$. We can partition the set $\{1, 2, \ldots, N\}$ into consecutive bands by considering the distinct floor values. Namely, if $\lfloor N/k \rfloor = \lfloor N/m \rfloor$ for $k, m \in \{1, 2, \ldots, N\}$, then k and m are elements of the same band. Being induced by the floor function, I call this partition, the Floor Band Decomposition. It is precisely this decomposition on the finite set of natural numbers $\{1, 2, \ldots, N\}$ that induces the Orthogonal Band Decomposition of finite D-matrices.

To study a signal, I consider its projections onto the orthogonal bands A_k. These projections are obtained via matrix multiplication by projection matrices P_k. In Lemma 4.4.1, I show that the matrices P_k are of the form

$$P_k = \lambda_k^{-1} A_k A_k^*. \tag{1.0.4}$$

The sum P of projection matrices P_k is an invertible matrix. In fact, in Theorem 4.4.3, I show that the matrix P factors as follows:

$$P = A \Lambda^{-1} A^*, \tag{1.0.5}$$

where Λ is the diagonal matrix with the norms λ_k along its diagonal. For a given D-matrix, I refer to the matrix P as the associated P-matrix.

The matrix P allows us to pass from representations of signals in the Fourier basis to representations in the basis induced by the projections P_k. In practice, signals often contain noise that we would like to remove. This requires that we modify the signals in a way that may result in loss of information. Approximating a signal with a partial sum of its projections onto the bands A_k, offers an approach to denoising. Preliminary experiments suggest that the error of approximating signals via partial sums of projections depends, significantly, on the choice of generating waveform f for the basis $\{f(nt)\}$. The hope is that the associated P-matrices may offer a more suitable metric to choose generating functions f in particular applications.

Chapter 2

Preliminaries

In this chapter I briefly mention a few mathematical tools for signal processing, which includes sampling, transforming and filtering of signals. This list is far from exhaustive; I only discuss methods which directly relate to the applications presented in this work. Specifically, I mention the Fourier representation of signals together with the Fast Fourier Transform (FFT), and the recently introduced D-bases together with their fast D-transforms. For a broader introduction to Signal Processing, I refer the reader to [9], [13], [14], [21], [29].

A note on notation: it is common in the literature to see a parameter t used to indicate continuous-time signals and a parameter n used to indicate digital time signals. Since my work focuses on digital signals, I did not find it necessary to make a special distinction in the way that I denote continuous and digital signals. Hence, for the sake of simplicity, I use the same parameter t to indicate both continuous and digital signals.

2.1 Digital Signals

While most signals are analog in nature (i.e., continuous functions), they become digital (discrete) when sampled at equal intervals. We can think of one-dimensional audio signals sampled at N time intervals, as vectors with N coordinates. In each coordinate we record the sampled amplitude of the signal in the corresponding time interval. Analogously, we can think of two-dimensional images, sampled over M equal intervals horizontally, and N equal intervals vertically, as rectangular matrices of size $M \times N$. The matrix entries a_{ij} record the grayscale of the image sample in the $(i,j)^{\text{th}}$ rectangle. On our screens, each little square is given by a pixel, i steps across and j steps up from the lower left corner.

The grayscale of each pixel is a number in the range $0 \leq a_{ij} < 256 = 2^8$, where 0 represents black and 255 represents white. The computer writes these numbers in binary notation, using 8 bits. For example, the all-white pixel 255 is given by 11111111 in binary notation, i.e., we have that

$$255 = \mathbf{1} \cdot 2^0 + \mathbf{1} \cdot 2^1 + \mathbf{1} \cdot 2^2 + \mathbf{1} \cdot 2^3 + \mathbf{1} \cdot 2^4 + \mathbf{1} \cdot 2^5 + \mathbf{1} \cdot 2^6 + \mathbf{1} \cdot 2^7. \tag{2.1.1}$$

Hence a grayscale image that has $M \cdot N$ pixels, with each pixel using 8 bits for its grayscale, becomes an $M \times N$ matrix with 256 possible values for each of its entries. This amounts to $8MN$ bits of information. For large enough M and N, this memory requirement can be significant. The need for compression becomes apparent.

With images, compression relies on the fact that changes in grayscale occur gradually, except for the edges. This means that nearby pixels, for the most part, have similar grayscale values. Edges, on the other hand, produce sudden jumps in grayscale values, making them hard to compress. In general, compression results in some loss of information. But if done right, there will be no noticeable discrepancy between the original and the compressed image.

Ultimately, the methods that we use to compress images, or signals in general, depend on the choice of basis representation for our signals. In fact, the choice of basis representation plays a key role in all aspects of signal processing (see [8], [30]) .

2.2 The Discrete Fourier Transform

This section gives a brief introduction to the Discrete Fourier Transform and the Fast Fourier Transform following the form of exposition used by Strang in [28]. To be able to discuss some potential signal processing applications of my work, I have found Strang's exposition to be particularly convenient. For a broader introduction to Fourier analysis and the Fourier Transforms, see [5], [13], [16], [18], [20], [32].

The Fourier basis allows us to represent a function $\phi \in L^2[0, 2\pi]$ as a sum of harmonics, i.e., $\phi(t) = \sum c_n e^{int}$. This is so because the sinusoidal functions $\{e^{int} \mid n \in \mathbb{Z}\}$ form an infinite complete collection of orthogonal functions on the interval $[0, 2\pi]$. The coefficients c_n represent the function in frequency space, while the values $\phi(t)$ represent the function in physical space. When working with finite data sets (digital signals), we can use the discrete time Fourier transform to go back and forth between the frequency representation and the physical representation of a function. The Fast Fourier Transform allows us to do that via a fast algorithm, speeding up the process from N^2 multiplications to $\frac{1}{2}N \log_2 N$ multiplications.

The key to the Discrete Fourier Transform is the Fourier matrix. The $N \times N$ Fourier matrix is generated by the N^{th} root of unity $w = w_N = e^{2\pi i/N}$, as follows:

$$F_N = \begin{bmatrix} 1 & 1 & 1 & \cdot & 1 \\ 1 & w & w^2 & \cdot & w^{N-1} \\ 1 & w^2 & w^4 & \cdot & w^{2(N-1)} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & w^{N-1} & w^{2(N-1)} & \cdot & w^{(N-1)^2} \end{bmatrix}. \tag{2.2.1}$$

Accepting the convention that the rows and columns of F_N are indexed by the integers from 0 to $N-1$ (instead of 1 to N), we see that the entry in the j^{th} row and the k^{th} column of F_N is w^{jk}. From here it is easy to see that the Fourier matrix F_N is symmetric (not Hermitian), that is, $F_N = F_N^T$. Furthermore, its columns are orthogonal. To see this, consider the inner product of the j^{th} row and the k^{th} column of F_N, with $j \neq k$:

$$(F_N \overline{F}_N)_{jk} = 1 + w^{j-k} + w^{2(j-k)} + \cdots + w^{(N-1)(j-k)} = \frac{1 - (w^{j-k})^N}{1 - w^{j-k}} = \frac{1 - (w^N)^{j-k}}{1 - w^{j-k}} = 0. \tag{2.2.2}$$

Note that, when $j = k$, we have that $(F_N \overline{F}_N)_{kk} = N$, i.e., we have that $F_N \overline{F}_N = NI$. It follows that $F_N^{-1} = \overline{F}_N/N$. We note that the orthogonality of the Fourier matrix plays a central role in the computations. Namely, having a fast algorithm for the forward transform automatically gives us a fast algorithm for the inverse transform as well.

Applying the Fourier transform is equivalent to multiplication by the Fourier matrix F_N:

$$F_N \mathbf{c} = \begin{bmatrix} 1 & 1 & 1 & \cdot & 1 \\ 1 & w & w^2 & \cdot & w^{N-1} \\ 1 & w^2 & w^4 & \cdot & w^{2(N-1)} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & w^{N-1} & w^{2(N-1)} & \cdot & w^{(N-1)^2} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \cdot \\ c_{N-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \cdot \\ y_{N-1} \end{bmatrix} = \mathbf{y}. \tag{2.2.3}$$

We apply the transform to the N complex coefficients $c_0, c_1, c_2, \ldots, c_{N-1}$, resulting in the N function values $y_0, y_1, y_2, \ldots, y_{N-1}$. The values $y_0, y_1, y_2, \ldots, y_{N-1}$ are the values of the finite Fourier series $\sum_0^{N-1} c_n e^{int}$ evaluated at the equally spaced points $t = 0, 2\pi/N, 4\pi/N, \ldots, 2(N-1)\pi/N$, on the interval $[0, 2\pi]$. That is,

$$y_j = c_0 + c_1 e^{i2j\pi/N} + c_2 e^{i4j\pi/N} + \cdots + c_{N-1} e^{ij(N-1)\pi/N} \tag{2.2.4}$$

$$= c_o + c_1 w^j + c_1 w^{2j} + \cdots + c_{N-1} w^{j(N-1)}. \tag{2.2.5}$$

Remark 2.2.1. *In MATLAB, the built-in Fourier transform fft(*$\mathbf{y}$*) gives the Fourier coefficients of the function. This is NOT the same as multiplication by* F_N*! The matrix* F_N *represents change of basis from "frequency space" to "physical space," that is, it reconstructs* $\mathbf{y}$ *from* $\mathbf{c}$*. Multiplication by the matrix* $\overline{F}_N$*, on the other hand, computes Fourier coefficients as fft(*$\mathbf{y}$*), i.e.,* $\mathbf{c} = F^{-1}\mathbf{y} = \frac{1}{N}\overline{F}\mathbf{y} = \frac{1}{N}$*fft(*$\mathbf{y}$*).*

Remark 2.2.2. *In this section I described how the Discrete Fourier Transform can be applied to transform one dimensional signals, i.e., vectors. The transform is extended to two dimensions, i.e., matrices, by repeatedly applying the one dimensional transform, first to the columns and then to the rows of the matrix. The MATLAB command is fft2(*$\mathbf{Y}$*). Similarly, the transform can be generalized to any finite dimension.*

2.2.1 The Matrix Factorization Behind the FFT

The goal is to multiply the vector $\mathbf{c}$ by the matrix F_N, as quickly as possible. Since the Fourier matrix has no zero entries, a direct multiplication would require N^2 separate multiplications. It is not obvious how multiplication by F_N can be carried out with fewer than N^2 multiplications. The idea behind the FFT is to take advantage of the special pattern $(F_N)_{jk} = w_N^{jk}$ to factor F_N in a way that produces many zeroes. The key idea is to connect F_N with the half-size Fourier matrix $F_{N/2}$, assuming that N is a power of 2.

When $N = 4$, the key is in the relation between F_4 and two copies of F_2:

$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & i^2 & i^3 \\ 1 & i^2 & i^4 & i^6 \\ 1 & i^3 & i^6 & i^9 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} & F_2 & \\ & & F_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & & \\ 1 & i^2 & & \\ & & 1 & 1 \\ & & 1 & i^2 \end{bmatrix}. \tag{2.2.6}$$

On the left we have F_4. On the right we have a matrix that is half zero. The two matrices are related through two additional sparse and simple matrices:

$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & i^2 & i^3 \\ 1 & i^2 & i^4 & i^6 \\ 1 & i^3 & i^6 & i^9 \end{bmatrix} = \begin{bmatrix} 1 & & 1 & \\ & 1 & & i \\ 1 & & -1 & \\ & 1 & & -i \end{bmatrix} \begin{bmatrix} 1 & 1 & & \\ 1 & i^2 & & \\ & & 1 & 1 \\ & & 1 & i^2 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & & 1 & \\ & 1 & & \\ & & & 1 \end{bmatrix}. \tag{2.2.7}$$

The rightmost matrix permutes the c's by putting the even c's (c_0 and c_2) ahead of the odd c's (c_1 and c_3). The middle matrix performs half-size transforms F_2 and F_2 on the even c's and odd c's. The matrix at the left combines the two half-size outputs in a way that produces the correct full-size output $\mathbf{y} = F_4 c$.

The same idea applies for any N, a power of 2. Set $M = \frac{1}{2}N$ and $w = e^{2\pi i/N}$. The Fourier matrix F_N contains powers of w. The first stage of the FFT is the factorization discovered by Cooley and Tukey:

$$F_N = \begin{bmatrix} I_M & D_M \\ I_M & -D_M \end{bmatrix} \begin{bmatrix} F_M & \\ & F_M \end{bmatrix} \begin{bmatrix} \text{even-odd} \\ \text{permutation} \end{bmatrix}, \tag{2.2.8}$$

where I_M is the identity matrix and D_M is the diagonal matrix with entries $1, w, \ldots, w^{M-1}$. Recall that the matrices F_M use the M^{th} root of unity, which is w^2. The permutation matrix separates the input vector $\mathbf{c}$ into its even and odd parts, $\mathbf{c}' = (c_0, c_2, \ldots, c_{N-2})$ and $\mathbf{c}'' = (c_1, c_3, \ldots, c_{N-1})$. Then the half sized transforms F_M are applied on $\mathbf{c}'$ and $\mathbf{c}''$ independently, giving $\mathbf{y}' = F_M \mathbf{c}'$ and $\mathbf{y}'' = F_M \mathbf{c}''$. Now applying the leftmost matrix yields $I\mathbf{y}' + D\mathbf{y}''$ for the first M components of $\mathbf{y}$ and $I\mathbf{y}' - D\mathbf{y}''$ for the last M components of $\mathbf{y}$. The formulas for each coordinate of $\mathbf{y}$ are:

$$y_j = y'_j + w^j y''_j, \quad j = 0, \ldots, M-1, \tag{2.2.9}$$

$$y_{j+M} = y'_j - w^j y''_j, \quad j = 0, \ldots, M-1. \tag{2.2.10}$$

As noted above, the key behind the factorization is $w^2 = w_M$, which in turn gives $w^{2jk} = w_M^{jk}$. The above formulas now come from separating $c_0, \ldots, c_{M-1}$ into even c_{2k} and odd c_{2k+1}, that is, for $\mathbf{y} = F_M \mathbf{c}$, we have that

$$y_j = \sum_{k=0}^{N-1} w^{jk} c_k = \sum_{k=0}^{M-1} w^{2jk} c_{2k} + \sum_{k=0}^{M-1} w^{j(2k+1)} c_{2k+1} \tag{2.2.11}$$

$$= \sum_{k=0}^{M-1} w_M^{jk} c_{2k} + w^j \sum_{k=0}^{M-1} w_M^{jk} c_{2k+1}. \tag{2.2.12}$$

Finally, note that for $j < M$ we have that $w_M^{(j+M)k} = w_M^{jk}$, while $w^{j+M} = w^j w^M = -w^j$ since $w^M = e^{\pi i} = -1$.

We reduced F_N to $F_{N/2}$. The next step is to reduce to $F_{N/4}$, then to $F_{N/8}$, and so on all the way to 1. Simply put, we can recursively apply the factorization of the Fourier matrix to each half-transform. This yields a fast algorithm. To see the improvement in speed, we count the number of multiplications that must be done at each step. For a Fourier matrix of size $N = 2^l$ there are $l = \log_2 N$ levels, going from $N = 2^l$ down

to $N = 1$. Each level has $N/2$ multiplications from the diagonal D_M's, to reassemble the half-size outputs from the lower level. The permutation matrices do not require multiplication, just a reordering. This yields the final count $\frac{1}{2}N \log_2 N$. An algorithm of order $\mathcal{O}(N \log N)$ is a fast algorithm.

We can use the FFT to compress a signal by removing all Fourier coefficients smaller than a set threshold. The figure below gives an example.

Figure 2.1: FFT thresholding

The original image on the left is 512×512 pixels. The compressed image was reconstructed using hard thresholding, preserving only the top 20% Fourier coefficients of the original image.

Remark 2.2.3. *The FFT factorization presented in this section is the simplest, radix-2, factorization for the Fourier matrix of size $N = 2^l$. Fourier matrices of an arbitrary composite size $N = N_1 N_2$ can be expressed through Fourier matrices of size N_1 and N_2 [6]. The applications, however, usually allow us to freely choose the size of the transform.*

2.2.2 Filtering in the Frequency Domain

A filter acts on a signal to produce a modified signal. In the physical domain, filtering amounts to convolution with a select kernel, which is not easy to do. In the frequency domain, on the other hand, filtering amounts to multiplication by a select window function. Furthermore, due to the orthogonality of the Fourier basis, the Fourier representation of a signal gives us decorrelated information about each frequency. It also allows us to alter one frequency without affecting the other frequencies. We can, therefore, define filters which allow for bands of frequencies to pass (altered or not), while stopping all other frequencies from passing. The simplest filter of this kind would be one where the lowest k many frequencies are allowed to pass unaltered, while the remaining frequencies are rejected. This would amount to multiplication by the diagonal matrix with 1's in the first k diagonal positions, and 0's everywhere else. As most noise (from measuring instruments) has high-frequency, this type of filter can be used to denoise. The filter will also remove fine features of the signal, such as the edges in images, while preserving the smooth features of the signal. This typically results in blurring.

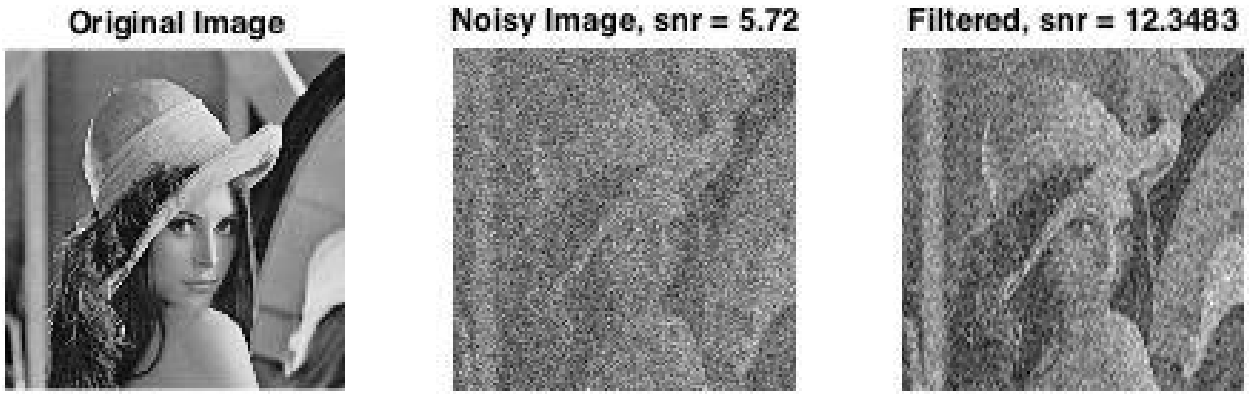

Figure 2.2: FFT low-pass filter

The original image on the left is 512×512 pixels. The noisy image was obtained by adding 80*randn(512) to the original image. The filter preserved the frequencies in the range $[-42, 42]$. The Signal to Noise Ratio (SNR) was computed using the MATLAB built-in snr(signal, noise) function.

2.3 Dirichlet Bases

In the foregoing section we discussed Fourier's idea to represent a function ϕ as a sum of harmonics

$$\phi(t) = \sum_n c_n e^{int}. \tag{2.3.1}$$

Note that the Fourier expansion of the function ϕ allows us to decompose it into a positive-frequency component, a zero-frequency component and a negative-frequency component via:

$$\phi(t) = \sum_{n=-\infty}^{\infty} c_n e^{int} = \sum_{n=-\infty}^{-1} c_n e^{int} + c_0 + \sum_{n=1}^{\infty} c_n e^{int} = \phi_{neg} + \phi_0 + \phi_{pos}. \tag{2.3.2}$$

This is a direct consequence of the orthogonality of the functions $\{e^{int}\}$ and it generalizes into a statement about the entire space $L^2[0, 2\pi]$. That is, the space $L^2[0, 2\pi]$ admits a direct decomposition

$$L^2[0, 2\pi] = H_a \oplus H_0 \oplus H_h, \tag{2.3.3}$$

where $H_a = \text{span}\{e^{-int} \mid n = 1, 2, 3, \dots\}$ is the negative-frequency (anti-holomorphic) subspace, $H_0 = \{1\}$ is the zero-fequency (trivial) subspace and $H_h = \text{span}\{e^{int} \mid n = 1, 2, 3, \dots\}$ is the positive-frequency (holomorphic) subspace of $L^2[0, 2\pi]$.

Representing functions as sums of harmonics (or sinusoidal waves) has proven very useful in signal processing, allowing us to study signals in frequency space via the FFT. But what about the idea of representing functions as sums of some other waveform $f(t)$ and its higher modes? Signal processing can often benefit from customized bases, especially if the associated transforms are computationally competitive.

As it turns out, there is an infinite family of bases for H_h of the form

$$\{f(nt) \mid n = 1, 2, 3 \dots\}, \tag{2.3.4}$$

generated by waveforms $f(t)$ and their higher modes. A basis $\{f(nt) \mid n = 1, 2, 3, \dots\}$ for H_h can be trivially extended to a basis $\{f(nt), \overline{f(nt)} \mid n = 1, 2, 3, \dots\} \cup \{1\}$ for the entire $L^2[0, 2\pi]$ space (the definition of the functions f_n below makes this obvious). As in the case of the FFT, the associated transforms are also fast. These bases, as well as their transforms, were recently introduced by Sowa (see [22], [24] and [27]). Some aspects of such bases were also investigated in preexisting engineering literature (see [10] and [11]). As we will see, in detail, in the subsequent chapter, these new transforms are closely related to the Dirichlet series. For this reason, I will refer to the bases $\{f(nt) \mid n = 1, 2, 3 \dots\}$ as Dirichlet bases, or D-bases for short. I will refer to their transforms as D-transforms.

The theory of the infinite D-bases and their transforms is presented in [27]. For my work, I am primarily interested in the discrete finite case. In this section I will assume that all series have been truncated and include only a finite number of terms. In particular, the indices will range over the numbers $1, 2, \dots, N$. More precisely, I will restrict my discussion to the finite positive-frequency subspace H_h^N whose elements are holomorphic functions sampled at N equal subintervals.

For a sequence $(\alpha_k)_{k=1}^N$ of complex numbers, the family of functions $\{f_m\}_{m=1}^N \subset H_h^N$ is defined in the following way:

$$f_m(t) = \sum_{k=1}^{\lfloor N/m \rfloor} \alpha_k e^{imkt}, \qquad m = 1, 2, \dots, N. \tag{2.3.5}$$

Modulo truncation, we have that $f_m(t) = f_1(mt)$ for all m. We set $f(t) = f_1(t) = \sum_{k=1}^N \alpha_k e^{ikt}$. Observe that $\alpha_1, \alpha_2, \dots, \alpha_N$ are the (positive-frequency) Fourier coefficients of f. This plays a very important role in the applications as it allows us to define a D-basis with a waveform of our choice and then use the Fourier coefficients of that waveform to apply the D-transform.

2.3.1 The D-transform

Let us consider the representation of a function $\phi \in H_h^N$ in the D-basis $\{f_n\}_{n=1}^N$ defined above. Then

$$\phi(t) = \sum_n x_n f_n(t) \tag{2.3.6}$$

$$= \sum_{n=1}^N \left(\sum_{k|n} \alpha_k x_{n/k} \right) e^{int} \tag{2.3.7}$$

$$= \sum_{n=1}^N y_n e^{int}. \tag{2.3.8}$$

We see that, for every $n = 1, 2, \dots, N$, the Fourier coefficients y_n of ϕ satisfy:

$$y_n = \sum_{k|n} \alpha_k x_{n/k}. \tag{2.3.9}$$

The first four Fourier coefficients, for example, satisfy:

$$y_1 = \alpha_1 x_1, \tag{2.3.10}$$

$$y_2 = \alpha_2 x_1 + \alpha_1 x_2, \tag{2.3.11}$$

$$y_3 = \alpha_3 x_1 + \alpha_1 x_3, \tag{2.3.12}$$

$$y_4 = \alpha_4 x_1 + \alpha_2 x_2 + \alpha_1 x_4. \tag{2.3.13}$$

The relation between the Fourier coefficients y_n and the D-basis coefficients x_n is given by

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ \cdot \\ y_N \end{bmatrix} = \begin{bmatrix} \alpha_1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \alpha_2 & \alpha_1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \alpha_3 & \cdot & \alpha_1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \alpha_4 & \alpha_2 & \cdot & \alpha_1 & \cdot & \cdot & \cdot & \cdot \\ \alpha_5 & \cdot & \cdot & \cdot & \alpha_1 & \cdot & \cdot & \cdot \\ \alpha_6 & \alpha_3 & \alpha_2 & \cdot & \cdot & \alpha_1 & \cdot & \cdot \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \cdot \\ \alpha_N & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ \cdot \\ x_N \end{bmatrix} = A\mathbf{x}. \tag{2.3.14}$$

The matrix A is sparse and completely defined by the sequence (α_k). The number of nontrivial entries in the n^{th} row is the number of divisors of n and multiplication by A can be carried out without non-trivial multiplications, resulting in a fast $\mathcal{O}(N \log N)$ algorithm. This algorithm, along with few other algorithms, will be presented in detail in Section 3.6.

Note that the invertibility of A depends solely on α_1 (the first mode of f), that is, A is invertible if and only if $\alpha_1 \neq 0$. When $\alpha_1 \neq 0$ we have that $\{f_n\}_{n=1}^N$ is a basis of H_h^N. The matrix A is the change of basis matrix from the D-basis representation to the Fourier basis representation and the columns of A feature the functions f_n, i.e.,

$$f_n = Ae_n, \tag{2.3.15}$$

where e_n is the n^{th} column of the identity matrix.

Significantly, the inverse of A has the same structure as A. That is,

$$A^{-1} = \begin{bmatrix} \gamma_1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \gamma_2 & \gamma_1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \gamma_3 & \cdot & \gamma_1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \gamma_4 & \gamma_2 & \cdot & \gamma_1 & \cdot & \cdot & \cdot & \cdot \\ \gamma_5 & \cdot & \cdot & \cdot & \gamma_1 & \cdot & \cdot & \cdot \\ \gamma_6 & \gamma_3 & \gamma_2 & \cdot & \cdot & \gamma_1 & \cdot & \cdot \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \cdot \\ \gamma_N & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \gamma_1 \end{bmatrix}. \tag{2.3.16}$$

We will see why this is true in Chapter 3. For now, it suffices to say that we have fast transforms to move back and forth between representations of signals in the Fourier basis and representations of signals in the D-bases. Furthermore, we only need to store the sequence (α_n) to apply the transforms. This, together with the FFT, gives us fast transforms to move back and forth between representations of signals in the physical space and representations of signals in the D-bases.

2.3.2 The Dual Basis

The Dual basis of $\{f_n\}_{n=1}^N$ is a set of functions $\{g_n\}_{n=1}^N$ on H_h^N satisfying

$$\langle f_n | g_m \rangle = \delta_{n,m}, \tag{2.3.17}$$

where $\delta_{n,m}$ is the Kronecker delta function. We will see that the functions g_n are defined by A^{-1}.

Define $g_n(t)$ as the function whose Fourier coefficients are the complex conjugates of the entries in the n^{th} row of A^{-1}, that is,

$$g_n(t) = \sum_{k|n} \overline{\gamma}_{\frac{n}{k}} e^{ikt}. \tag{2.3.18}$$

Equivalently,

$$g_n = (A^{-1})^* e_n. \tag{2.3.19}$$

From here, we have that

$$\langle f_n | g_m \rangle = \langle A e_n | (A^{-1})^* e_m \rangle = \langle A A^{-1} e_n | e_n \rangle = \langle e_n | e_m \rangle = \delta_{m,n}. \tag{2.3.20}$$

Given a function $\phi(t) = \sum_n x_n f_n(t) \in H_h^N$, we can use the dual basis to determine the D-basis coefficients x_n of ϕ. Namely, we have that

$$\langle \phi | g_n \rangle = \langle \sum_m x_m f_m | g_n \rangle = \sum_m x_m \langle f_m | g_n \rangle = x_n, \tag{2.3.21}$$

and hence

$$\phi(t) = \sum_n \langle \phi | g_n \rangle f_n(t). \tag{2.3.22}$$

2.3.3 Filtering in a D-basis

Filtering signals in a D-basis representation is similar to filtering in the Fourier basis representation. Let us consider a D-basis $\{f_n\}_{n=1}^N$ for H_h^N and a function $\phi \in H_h^N$. Recall that the D-basis expansion of ϕ is given by

$$\phi(t) = x_1 f(t) + x_2 f(2t) + x_3 f(3t) + \cdots + x_N f(Nt), \tag{2.3.23}$$

that is, we have ϕ represented as a sum of the generating waveform $f(t)$ and its higher modes. We see that this representation generalizes the Fourier representation. In other words, the notion of frequency from the Fourier basis continues to exist in the D-basis (we are still in the frequency domain). The key difference

lies in the fact that, unless $f(t)$ is a multiple of the first harmonic e^{it}, the set $\{f_n\}_{n=1}^{N}$ is not an orthogonal set (this is a direct consequence of Corollary 3.3.4). Hence, we cannot expect that changing one frequency response (coefficient x_n) will not affect the remaining frequencies.

The topic of filtering in a D-basis representation has motivated much of the research presented in this thesis. I will revisit this topic in Chapter 5.

Chapter 3

Dirichlet Matrices

In Section 2.3, we saw that D-matrices arise as change of basis transforms when we consider bases of the form $\{f(nt) \mid n = 1, 2, 3, \dots\}$. To be able to advance D-transforms towards signal processing applications, it is important that we understand the mathematical properties of D-matrices. As my work focuses on the finite case, this chapter offers an in-depth algebraic introduction into D-matrices, including the isomorphism with Dirichlet series, computational compatibility with finite truncations, the Hermitian adjoint and its properties, the properties of the Hermitian matrices $AA^\star$ and $A^\star A$, fast algorithms for computations with D-matrices, and finally actions with D-matrices.

While some of the results of this chapter might not be directly applicable in the context of signal processing, I have included them so as to offer a more rounded presentation of the general algebraic theory of D-matrices. With the intention of making my work accessible to a broader audience, within and outside of mathematics, in this chapter I have taken a new, basis approach, using elementary D-matrices, to introduce the algebraic theory of D-matrices. I believe that elementary D-matrices are helpful in making some of the proofs more lucid, in particular to readers who are not experts in Linear Algebra.

3.1 The Set of Dirichlet series

Dirichlet series have a long history in mathematics (see [2], [4], [12]). In this section we recap some of the basic properties that we need to establish the relationship between Dirichlet series and D-matrices.

A formal Dirichlet series is a series of the form

$$A(s) = \sum_{n=1}^{\infty} \frac{a_n}{n^s}, \tag{3.1.1}$$

where (a_n) is a sequence of complex numbers and s is a complex variable. The set of Dirichlet series admits the operations of addition and multiplication, with the sum of two Dirichlet series given by

$$\sum_{n=1}^{\infty} \frac{a_n}{n^s} + \sum_{n=1}^{\infty} \frac{b_n}{n^s} = \sum_{n=1}^{\infty} \frac{a_n + b_n}{n^s}, \tag{3.1.2}$$

and the product given by

$$\left(\sum_{n=1}^{\infty} \frac{a_n}{n^s}\right) \cdot \left(\sum_{n=1}^{\infty} \frac{b_n}{n^s}\right) = \sum_{n=1}^{\infty} \frac{c_n}{n^s}, \text{ with } c_n = \sum_{d|n, d\geq 1} a_d \cdot b_{n/d}. \tag{3.1.3}$$

We note that the Dirichlet series representation of 1 is

$$1 = \frac{1}{1^s} + \frac{0}{2^s} + \frac{0}{3^s} + \dots. \tag{3.1.4}$$

From here we can see that a Dirichlet series has a multiplicative inverse if and only if its first term is non-zero. Given an invertible Dirichlet series

$$\sum_{n=1}^{\infty} \frac{a_n}{n^s}, \text{ with } a_1 \neq 0, \tag{3.1.5}$$

we can, by inspection, confirm that its Dirichlet inverse is defined by the sequence (b_n), with

$$b_1 = \frac{1}{a_1} \text{ and } b_n = -\frac{1}{a_1} \sum_{d|n,\ d>1} b_{n/d} \cdot a_d, \text{ for } n > 1. \tag{3.1.6}$$

With the operations of addition and multiplication, the set of formal Dirichlet series forms a commutative ring. I will refer to this ring as the Dirichlet ring.

Consider the set of elementary sequences $\{(a_n = \delta_{nk}) \mid k \in \mathbb{N}\}$, where $\mathbb{N}$ denotes the set of positive natural numbers. I will refer to the corresponding set of Dirichlet series

$$\left\{ \frac{1}{n^s} \mid n \in \mathbb{N} \right\} \tag{3.1.7}$$

as the set of elementary Dirichlet series. Note that the product of two elementary Dirichlet series is also an elementary Dirichlet series, that is,

$$\frac{1}{m^s} \cdot \frac{1}{n^s} = \frac{1}{(mn)^s}. \tag{3.1.8}$$

Consequently, we have that the set of elementary Dirichlet series is a commutative multiplicative monoid.

3.2 Elementary D-matrices

In this section I introduce an alternative representation to elementary Dirichlet series by infinite sparse lower triangular matrices. I will refer to such matrices as elementary D-matrices. I will show that the set of elementary Dirichlet series admits an isomorphism of monoids to the set of elementary D-matrices.

For every natural number n, define the **elementary D-matrix** D_n to be the infinite matrix with terms

$$(D_n)_{ij} = \begin{cases} 1 & \text{if } i = nj; \\ 0 & \text{otherwise.} \end{cases} \tag{3.2.1}$$

The elementary D-matrices corresponding to $n = 1, 2, 3$ and 4, for example, exhibit the following structure:

$$D_1 = \begin{bmatrix} 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \quad D_2 = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

$$D_3 = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \quad D_4 = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

Lemma 3.2.1. *The set of elementary D-matrices, together with the operation of matrix multiplication, is a commutative monoid, isomorphic to the set of elementary Dirichlet series.*

Proof. As remarked above, D_1 is the identity matrix, implying that the set of elementary D-matrices has a multiplicative identity. Next, take two elementary D-matrices, D_m and D_n, and consider their product $D_m D_n$. A non-zero term of the matrix $D_m D_n$ must be the dot product of a non-zero row in D_m and a non-zero column in D_n. Specifically, if $(D_m D_n)_{ij} = 1$, then there exists a $k \in \mathbb{N}$ such that $i = mk$ and $k = nj$. This implies that

$$(D_m D_n)_{ij} = \begin{cases} 1 & \text{if } i = (mn)j \text{ and ;} \\ 0 & \text{otherwise.} \end{cases} \tag{3.2.2}$$

This is, by definition, D_{mn}. Thus it follows that

$$D_m D_n = D_{mn}. \tag{3.2.3}$$

Furthermore, since multiplication of natural numbers is commutative, we have that the product of two elementary D-matrices is also commutative. Hence we have shown that the set of elementary D-matrices forms a commutative monoid.

Finally, the map

$$\iota \ : \ \frac{1}{n^s} \longmapsto D_n \tag{3.2.4}$$

is the desired isomorphism from the set of elementary Dirichlet series to the set of elementary D-matrices. □

3.3 The Set of D-matrices

In the previous section I established the existence of the isomorphism $\iota : n^{-s} \mapsto D_n$, from the set of elementary Dirichlet series, to the set of elementary D-matrices. In this section, I will consider an extension $\tilde{\iota}$ of ι. I will show that this extension is an injective ring homomorphism from the Dirichlet ring to the set of infinite matrices, and hence, an isomorphism onto its range.

Theorem 3.3.1. *The map*

$$\tilde{\iota} \ : \ \sum_{n=1}^{\infty} \frac{a_n}{n^s} \longmapsto \sum_{n=1}^{\infty} a_n D_n, \tag{3.3.1}$$

from the Dirichlet ring, into the set of infinite dimensional matrices, is a ring homomorphism.

Proof. Let $\sum_{n=1}^{\infty} a_n n^{-s}$ and $\sum_{n=1}^{\infty} b_n n^{-s}$ be two Dirichlet series. Matrix addition implies that $\tilde{\iota}$ is additive. That is,

$$\tilde{\iota}\left(\sum_{n=1}^{\infty} \frac{(a_n + b_n)}{n^s}\right) = \sum_{n=1}^{\infty}(a_n + b_n)D_n = \sum_{n=1}^{\infty} a_n D_n + \sum_{n=1}^{\infty} b_n D_n = \tilde{\iota}\left(\sum_{n=1}^{\infty} \frac{a_n}{n^s}\right) + \tilde{\iota}\left(\sum_{n=1}^{\infty} \frac{b_n}{n^s}\right). \tag{3.3.2}$$

Next we will see that $\tilde{\iota}$ respects multiplication. Let $\sum_{n=1}^{\infty} c_n n^{-s} = \sum_{n=1}^{\infty} a_n n^{-s} \cdot \sum_{n=1}^{\infty} b_n n^{-s}$. Recall that the elements of the sequence (c_n) are given by $c_n = \sum_{d|n, d\geq 1} a_d \cdot b_{n/d}$. Recall also, that the product of two elementary D-matrices is also an elementary D-matrix, that is, $D_m D_n = D_{mn}$. From here we have that

$$\sum_{n=1}^{\infty} c_n D_n = \sum_{n=1}^{\infty}\left(\sum_{d|n, d\geq 1} a_d \cdot b_{n/d}\right) D_n = \sum_{n=1}^{\infty}\sum_{d|n, d\geq 1} a_d D_d \cdot b_{n/d} D_{n/d} = \sum_{n=1}^{\infty} a_n D_n \cdot \sum_{n=1}^{\infty} b_n D_n, \tag{3.3.3}$$

which in turn implies that

$$\tilde{\iota}\left(\sum_{n=1}^{\infty} \frac{a_n}{n^s} \cdot \sum_{n=1}^{\infty} \frac{b_n}{n^s}\right) = \tilde{\iota}\left(\sum_{n=1}^{\infty} \frac{a_n}{n^s}\right)\tilde{\iota}\left(\sum_{n=1}^{\infty} \frac{b_n}{n^s}\right). \tag{3.3.4}$$

It remains to show that $\tilde{\iota}$ is injective. Take two D-series $\sum_{n=1}^{\infty} a_n n^{-s}$ and $\sum_{n=1}^{\infty} b_n n^{-s}$. If

$$\sum_{n=1}^{\infty} a_n D_n = \tilde{\iota}\left(\sum_{n=1}^{\infty} a_n n^{-s}\right) = \tilde{\iota}\left(\sum_{n=1}^{\infty} b_n n^{-s}\right) = \sum_{n=1}^{\infty} b_n D_n, \tag{3.3.5}$$

then we have that

$$\sum_{n=1}^{\infty}(a_n - b_n)D_n = 0, \tag{3.3.6}$$

which implies that $a_n = b_n$ for every $n \in \mathbb{N}$. □

Let $A = \sum_{n=1}^{\infty} a_n D_n$. We refer to matrices of this form as **Dirichlet matrices** or **D-matrices.** Observe that the structure of A is as follows

$$A = \begin{bmatrix} a_1 & & & & & & & & \cdots \\ a_2 & a_1 & & & & & & & \cdots \\ a_3 & & a_1 & & & & & & \cdots \\ a_4 & a_2 & & a_1 & & & & & \cdots \\ a_5 & & & & a_1 & & & & \cdots \\ a_6 & a_3 & a_2 & & & a_1 & & & \cdots \\ a_7 & & & & & & a_1 & & \cdots \\ a_8 & a_4 & & a_2 & & & & a_1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{3.3.7}$$

Note that the generating sequence (a_n) is given by, or featured in, the first column of A, that is, $a_n = A_{n1}$. More precisely, the entries of A are given by

$$A_{ij} = \begin{cases} a_{i/j} & \text{if } j \mid i \text{ } (j \text{ divides } i); \\ 0 & \text{otherwise.} \end{cases} \tag{3.3.8}$$

Having defined the set of D-matrices as the range of the ring homomorphism $\tilde{\iota}$, we have the following results.

Corollary 3.3.2. *The set of D-matrices is a commutative ring, isomorphic to the Dirichlet ring.*

Corollary 3.3.3. *Let $A = \sum_{n=1}^{\infty} a_n D_n$ be a D-matrix. A is invertible if and only if $a_1 \neq 0$. If A is invertible, then its inverse is the D-matrix $B = \sum_{n=1}^{\infty} b_n D_n$, with*

$$b_1 = \frac{1}{a_1} \text{ and } b_n = -\frac{1}{a_1} \sum_{d|n,\ d>1} b_{n/d} \cdot a_d, \text{ for } n > 1. \tag{3.3.9}$$

Corollary 3.3.4. *Let $A = \sum_{n=1}^{\infty} a_n D_n$ be a D-matrix. A is unitary if and only if $A = a_1 I$ with $|a_1| = 1$.*

Proof. Suppose that A is a unitary matrix. Then A^{-1} is equal to the conjugate transpose of A. Since A^{-1} is a D-matrix, it follows that A^{-1} is lower triangular. On the other hand, since A^{-1} coincides with the conjugate transpose of A, it follows that A^{-1} is upper triangular as well. This implies that A^{-1} is a diagonal matrix, which, in turn, implies that A is a diagonal matrix. Since the diagonal of A features a_1 and A is unitary, it follows that $a_1 \overline{a}_1 = 1$. □

Recall that D-matrices represent change of basis transformations between representations of functions in the Fourier basis and representations of functions in a D-basis. By virtue of Corollary 3.3.4 we know that the only orthogonal D-basis is in fact the Fourier basis (upto multiplication by constants).

3.4 The Finite D-matrix

While the theory of infinite matrices is substantially different from the theory of finite matrices, that is, infinite matrices are studied in Analysis, while finite ones are studied in Linear Algebra, one often studies

infinite objects by considering their finite counterparts. We will see, in the subsequent proposition, that one can approximate a D-matrix by considering the matrix obtained by truncating to the upper-left corner of size $N \times N$.

Proposition 3.4.1. *Let $A = \sum_{n=1}^{\infty} a_n D_n$ and $B = \sum_{n=1}^{\infty} b_n D_n$ be two (infinite) D-matrices. Let $A_{\lrcorner N}$ denote the matrix A truncated to the upper left corner of size $N \times N$ and let $B_{\lrcorner N}$ denote the corresponding truncated matrix B. Then we have that*

$$A_{\lrcorner N} + B_{\lrcorner N} = (A + B)_{\lrcorner N} \tag{3.4.1}$$

and

$$A_{\lrcorner N} B_{\lrcorner N} = (AB)_{\lrcorner N}, \tag{3.4.2}$$

where $(A + B)_{\lrcorner N}$ and $(AB)_{\lrcorner N}$ denote the D-matrices $A + B$ and AB truncated to the upper left corners of size $N \times N$, respectively.

Proof. The definition of matrix addition implies, trivially, that the equality $A_{\lrcorner N} + B_{\lrcorner N} = (A + B)_{\lrcorner N}$ holds. Recall that the term in the ij^{th} position of the product matrix AB is defined as the inner product of the i^{th} row of A and the j^{th} column of B. Since A is lower triangular, the non-zero terms in the i^{th} row are located within the first i coordinates. Hence the inner product of the i^{th} row of A and the j^{th} column of B involves, non-trivially, no more than the first i elements of the j^{th} column of B. Hence the product of two D-matrices is compatible with truncation to the upper left corner of size $N \times N$. □

Henceforth, I will refer to truncated D-matrices as **finite D-matrices.** Observe that the elementary D-matrices are infinite dimensional matrices. Hence, note that the finite sum $\sum_{n=1}^{N} a_n D_n$ is not a finite D-matrix of size $N \times N$, but rather a finitely generated infinite D-matrix. To specify the generating sequence of a finite D-matrix, however, I will often use the notation $\sum_{n=1}^{N} a_n D_n$ with the implication that the elementary matrices $D_1, \ldots, D_N$ have been truncated to the upper left corners of size $N \times N$. The use of this notation will be made clear from the context.

Proposition 3.4.2. *Let D_k be the k^{th} elementary D-matrix of size $N \times N$. Then*

$$rank(D_k) = \left\lfloor \frac{N}{k} \right\rfloor. \tag{3.4.3}$$

Proof. By definition, every non-zero row of D_k must be indexed by a multiple of k. The number of multiples of k, bounded above by N, is given by $\lfloor N/k \rfloor$. □

Proposition 3.4.3. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite D-matrix of size $N \times N$. Then the k^{th} column of A features the first $\lfloor N/k \rfloor$ terms of the generating sequence (a_n) of A.*

Proof. Recall that the entries of A are given by

$$A_{ij} = \begin{cases} a_{i/j} & \text{if } j \mid i \text{ } (j \text{ divides } i); \\ 0 & \text{otherwise.} \end{cases} \tag{3.4.4}$$

Hence the entries of the k^{th} column, which are not trivially zero, are $A_{kk}, A_{2k,k}, \ldots, A_{\lfloor kN/k \rfloor,k}$. These terms correspond to $a_1, a_2, \ldots, a_k$, respectively. □

Proposition 3.4.4. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite D-matrix of size $N \times N$. Then the k^{th} row of A features $d(k)$ many elements of the generating sequence (a_n) of A, where $d(k)$ is the number of divisors of k.*

Proof. Consider the k^{th} row of A. For every positive divisor l of k we have that the entry $A_{k,k/l} = a_{k/l}$, while all other terms are equal to zero. □

Observe that the foregoing propositions show the following number theoretic fact

$$\sum_{k=1}^{N} \left\lfloor \frac{N}{k} \right\rfloor = \sum_{k=1}^{N} d(k). \tag{3.4.5}$$

Furthermore, they show that evaluating a finite D-matrix at an N- dimensional vector requires $\sum_{k=1}^{N} \lfloor N/k \rfloor$ non-trivial operations, rather than the usual N^2 operations necessary to evaluate a square matrix of size $N \times N$. It is well known that $\sum_{k=1}^{N} \lfloor N/k \rfloor$ is of order $\mathcal{O}(N \log N)$, a computational complexity considered to be fast. (A more precise estimate is given in [23]). Hence, algorithms which avoid the trivial zeros of a D-matrix would be considered fast, and therefore numerically advantageous. In Section 3.6 I present a number of algorithms, including an algorithm for evaluating a D-matrix at a vector (or a matrix), an algorithm for computing the inverse of a D-matrix and an algorithm to evaluate the product of two D-matrices. I will also show that these algorithms are indeed of complexity $\mathcal{O}(N \log N)$, and are therefore computationally desirable.

3.5 The Hermitian Adjoint of the D-Matrix

Let $A = \sum_{n=1}^{\infty} a_n D_n$ be a D-matrix. I will denote by A^* the Hermitian adjoint (or conjugate transpose) of A, that is

$$A^* = \sum_{n=1}^{\infty} \overline{a}_n D_n^T. \tag{3.5.1}$$

Note that the entries of ${D_n}^T$ satisfy

$$({D_n}^T)_{ij} = \begin{cases} 1 & \text{if } j = ni; \\ 0 & \text{otherwise.} \end{cases} \tag{3.5.2}$$

The product $D_m {D_n}^T$ is given by

$$(D_m {D_n}^T)_{ij} = \begin{cases} 1 & \text{if } (m,n) = \frac{1}{k}(i,j) \text{ for } k \in \mathbb{N}; \\ 0 & \text{otherwise,} \end{cases} \tag{3.5.3}$$

while the product ${D_m}^T D_n$ is given by

$$({D_m}^T D_n)_{ij} = \begin{cases} 1 & \text{if } (m,n) = k(\frac{L}{i}, \frac{L}{j}) \text{ for } k \in \mathbb{N} \text{ and } L = \operatorname{lcm}(i,j); \\ 0 & \text{otherwise.} \end{cases} \tag{3.5.4}$$

Proposition 3.5.1. *Let $A = \sum_{n=1}^{\infty} a_n D_n$ be a D-matrix. Let $d = gcd(i,j)$ be the greatest common divisor of i and j. Then the entries of the Hermitian matrix AA^* are given by*

$$(AA^*)_{ij} = \sum_{l|d} a_{i/l}\bar{a}_{j/l}. \tag{3.5.5}$$

Proof. Note that

$$AA^* = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} a_m \bar{a}_n D_m {D_n}^T. \tag{3.5.6}$$

Consider $(AA^*)_{ij}$, for some fixed i and j. Then we have that,

$$(AA^*)_{ij} = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} a_m \bar{a}_n (D_m {D_n}^T)_{ij}. \tag{3.5.7}$$

As noted above, the product $(D_m {D_n}^T)_{ij} \neq 0$ if and only if $(m,n) = \frac{1}{k}(i,j)$ for $k \in \mathbb{N}$. Since $m, n \in \mathbb{N}$, if $(D_m {D_n}^T)_{ij} \neq 0$, then k must divide both i and j and hence, k must divide $d = \gcd(i,j)$. For every positive divisor l of d, we have that

$$(D_{i/l} {D_{j/l}}^T)_{ij} = 1. \tag{3.5.8}$$

For all other m and n, we have that $(D_m D_n^T)_{ij} = 0$. From here the result follows. □

Proposition 3.5.2. *Let $A = \sum_{n=1}^{\infty} a_n D_n$ be a D-matrix. Then the Hermitian matrix AA^* is compatible with truncation to the upper left corner of size $N \times N$. Specifically, we have that*

$$(AA^*)_{\lrcorner N} = A_{\lrcorner N} A^*_{\lrcorner N}. \tag{3.5.9}$$

Proof. Observe that the sum

$$(AA^*)_{ij} = \sum_{l|d} a_{i/l}\bar{a}_{j/l} \tag{3.5.10}$$

involves terms of the generating sequence with indices bounded by i and j, that is, bounded by the maximum of the two. Hence the term $(AA^*)_{ij}$, for $i, j \leq N$ is entirely determined by the first N elements of the generating sequence (a_n). This implies that the matrix AA^* can be approximated with the finite matrices obtained by truncation to the upper left corner of size $N \times N$. □

Proposition 3.5.3. *Let $A = \sum_{n=1}^{\infty} a_n D_n$ be a D-matrix. Let $L = lcm(i,j)$ be the least common multiple of i and j. Then the entries of the Hermitian matrix A^*A are given by*

$$(A^*A)_{ij} = \sum_{k=1}^{\infty} \bar{a}_{k\frac{L}{i}} a_{k\frac{L}{j}}. \tag{3.5.11}$$

Proof. For fixed i and j we have that

$$(A^*A)_{ij} = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} \bar{a}_m a_n ({D_m}^T D_n)_{ij}. \tag{3.5.12}$$

Recall that $({D_m}^T D_n)_{ij} \neq 0$ if and only if $(m,n) = k(\frac{L}{i}, \frac{L}{j})$ for some $k \in \mathbb{N}$. Since $L = \text{lcm}(i,j)$, we have that $i \mid L$ and $j \mid L$. This means that $\frac{L}{i}, \frac{L}{j} \in \mathbb{N}$, which in turn implies that there are no restrictions on $k \in \mathbb{N}$. □

Proposition 3.5.4. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite D-matrix. Let $L = lcm(i,j)$ denote the least common multiple of i and j. If $L = lcm(i,j) > N$ then we have that $(A^*A)_{ij} = 0$. Alternatively, if $L \leq N$, then the entries of the finite Hermitian matrix A^*A are given by*

$$(A^*A)_{ij} = \sum_{k=1}^{\lfloor N/L \rfloor} \bar{a}_{k\frac{L}{i}} a_{k\frac{L}{j}}. \tag{3.5.13}$$

Proof. Recall that $(A^*A)_{ij}$ gives the inner product of the i^{th} and j^{th} columns of A. Hence

$$(A^*A)_{ij} = \sum_{t=1}^{N} \bar{A}_{ti} A_{tj} = \sum_{i|t, j|t} \bar{a}_{\frac{t}{i}} a_{\frac{t}{j}} = \sum_{L|t} \bar{a}_{\frac{t}{i}} a_{\frac{t}{j}}. \tag{3.5.14}$$

If $L > N$, clearly $(A^*A)_{ij} = 0$, and the corresponding columns are orthogonal. If $L \leq N$, then we have that $1 \leq \frac{t}{i} \leq \frac{N}{i}$. Since $L \mid t$, we have that $t = kL$, for some $k \in \mathbb{N}$, which implies that

$$1 \leq k\frac{L}{i} \leq \frac{N}{i}. \tag{3.5.15}$$

From here we have that $k \leq \lfloor N/L \rfloor$ and that

$$(A^*A)_{ij} = \sum_{L|k} \bar{a}_{\frac{k}{i}} a_{\frac{k}{j}} = \sum_{k=1}^{\lfloor N/L \rfloor} \bar{a}_{k\frac{L}{i}} a_{k\frac{L}{j}}. \tag{3.5.16}$$

□

Corollary 3.5.5. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite D-matrix. Take $i, j \in \{1, 2, \ldots, N\}$. If $lcm(i,j) > N$, then the i^{th} and j^{th} columns of A are mutually orthogonal.*

3.6 Computations with D-matrices

In this section I present a number of fast algorithms involving finite D-matrices and their Hermitian adjoints. Algorithms 3.6.2, 3.6.6 and 3.6.9 have been introduced in [23]. Here I present them in a modified form along with detailed explanations. Algorithms 3.6.4 and 3.6.10 are generalizations of Algorithms 3.6.2 and 3.6.9, respectively. All algorithms are presented in MATLAB syntax.

Proposition 3.6.1. *Let A be a D-matrix of size $N \times N$ and let v be an N- dimensional vector. Then $u = Av$ can be computed via a fast algorithm of order $\mathcal{O}(N \log N)$, with memory requirement of order $\mathcal{O}(N)$.*

Proof. Recall that a D-matrix is uniquely determined by its first column. Let $a = [a(1)\ a(2)\ \ldots\ a(N)]^T$ be the first column of A, that is, let $a(k) = A_{k1}$ for $k = 1, 2, \ldots, N$. If $v(1),\ v(2),\ \ldots,\ v(N)$ denote the

coordinates of v, then the product Av is given by

$$\begin{bmatrix} u(1) \\ u(2) \\ u(3) \\ u(4) \\ u(5) \\ u(6) \\ \vdots \\ u(N) \end{bmatrix} = v(1) \begin{bmatrix} a(1) \\ a(2) \\ a(3) \\ a(4) \\ a(5) \\ a(6) \\ \vdots \\ a(N) \end{bmatrix} + v(2) \begin{bmatrix} \cdot \\ a(1) \\ \cdot \\ a(2) \\ \cdot \\ a(3) \\ \cdot \\ \vdots \end{bmatrix} + \cdots + v(N) \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ a(1) \end{bmatrix}. \tag{3.6.1}$$

Observe that, in the sum above, $v(1)$ multiplies every coordinate of a, $v(2)$ multiplies the first $\lfloor N/2 \rfloor$ coordinates of a, $v(3)$ multiplies the first $\lfloor N/3 \rfloor$ coordinates of a, and so on. In general, $v(k)$ multiplies the first $\lfloor N/k \rfloor$ coordinates of a, and it affects every k^{th} term of the sum. Hence, the product Av may be implemented via the following algorithm.

Algorithm 3.6.2. *For input vectors v and a, initiate $u = v(1) * a$;*

for $k = 2 : N$

$u(k : k : N) = u(k : k : N) + v(k) * a(1 : floor(N/k))$;

end

where $u(k : k : N)$ is the vector consisting of every k^{th} element of u.

Observe that at the k^{th} iteration of the loop, there are $2 \cdot \lfloor N/k \rfloor$ elementary operations. The total number of operations is

$$N + 2\left(\sum_{k=2}^{N} \left\lfloor \frac{N}{k} \right\rfloor\right) = \mathcal{O}(N) + \mathcal{O}(N \log N) = \mathcal{O}(N \log N). \tag{3.6.2}$$

Since the complexity of the algorithm is of order $\mathcal{O}(N \log N)$, this algorithm is considered to be a fast algorithm. Furthermore, since the algorithm requires only the generating sequence (a_n) (rather than the matrix A), the memory requirement is of order $\mathcal{O}(N)$. □

The above algorithm can be generalized to matrices (rather than vectors) of size $N \times M$.

Proposition 3.6.3. *Let A be a D-matrix of size $N \times N$ and let V be a matrix of size $N \times M$. Then the product $U = AV$ can be computed via an algorithm of order $\mathcal{O}(NM \log N)$, with memory requirement of order $\mathcal{O}(NM)$.*

Proof. Let $V_1, V_2, \ldots, V_M$ represent the columns of V, and let $U_1, U_2, \ldots, U_M$ represent the columns of U. Recall that $U_1 = AV_1,\ U_2 = AV_2, \ldots, U_M = AV_M$. Hence, each column of U can be computed via Algorithm 3.6.2. Note that the k^{th} iteration of the loop in Algorithm 3.6.2 alters each k^{th} row of the output. Analogously, we can compute the product AV via an algorithm that alters each k^{th} row of the output at the k^{th} iteration. Namely, Algorithm 3.6.2 admits the following generalization.

Algorithm 3.6.4. *For input matrix U and vector a, initiate $U = zeros(size(V))$;*

$for\ k = 1 : N$

$\quad U(k : k : N, :) = U(k : k : N, :) + a(1 : floor(N/k)) * V(k, :);$

end

Since for each $k = 1, 2, \ldots, N$, the number of columns of $U(k : k : N, :)$ is M, the total number of operations is $\mathcal{O}(MN \log N)$. □

Similarly, the inverse of a D-matrix can be computed via a fast algorithm directly from the generating sequence.

Proposition 3.6.5. *Let A be a D-matrix of size $N \times N$. Then A^{-1} can be computed via a fast algorithm of order $\mathcal{O}(N \log N)$, with memory requirement of order $\mathcal{O}(N)$.*

Proof. Let $a = [a(1)\ a(2)\ \ldots\ a(N)]^T$ denote the first column of A and let $b = [b(1)\ b(2)\ \ldots\ b(N)]^T$ denote the first column of A^{-1}. Recall that the coordinates of b are given by

$$b(1) = \frac{1}{a(1)} \text{ and } b(k) = -b(1) \sum_{d|k,\ d>1} b(k/d) \cdot a(d), \text{ for } k = 2, \ldots, N. \tag{3.6.3}$$

If we set $b(1) = 1/a(1)$, then b can be represented as

$$\begin{bmatrix} b(1) \\ b(2) \\ b(3) \\ b(4) \\ b(5) \\ b(6) \\ \vdots \\ b(N) \end{bmatrix} = -b(1)^2 \begin{bmatrix} -a(1) \\ a(2) \\ a(3) \\ a(4) \\ a(5) \\ a(6) \\ \vdots \\ a(N) \end{bmatrix} - b(1)b(2) \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ a(2) \\ \cdot \\ a(3) \\ \cdot \\ \vdots \end{bmatrix} - \cdots - b(1)b(\lfloor \tfrac{N}{2} \rfloor) \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \vdots \\ a(2) \\ \vdots \end{bmatrix}. \tag{3.6.4}$$

The above sum can be computed without trivial operations (i.e., adding/multiplying by 0) via the following fast algorithm.

Algorithm 3.6.6. *For input vector a, compute output vector b:*

$b = (-1/a(1)^2) * a;$

$b(1) = -b(1);$

$for\ k = 2 : floor(N/2)$

$\quad b(2 * k : k : N) = b(2 * k : k : N) - (b(1) * b(k)) * a(2 : floor(N/k));$

end

Observe that the k^{th} iteration of the for-loop requires $2 \cdot \lfloor N/k \rfloor - 1$ basic operations. The total number of basic operations is therefore

$$(3+N)+1+\sum_{k=2}^{\lfloor N/2 \rfloor}(1+2(\lfloor N/k \rfloor - 1)) = 2\sum_{k=2}^{\lfloor N/2 \rfloor}\lfloor N/k \rfloor + \mathcal{O}(N) = \mathcal{O}(N \log N). \tag{3.6.5}$$

Again, since A^{-1} is entirely determined by its first column, and since the computation of A^{-1} requires only the generating sequence of A, the memory requirements for this algorithm are of order $\mathcal{O}(N \log N)$. $\square$

From the following proposition we see that the product of two D-matrices can also be computed via a fast algorithm which requires only the generating sequences.

Proposition 3.6.7. *Let A and B be two D-matrices of size $N \times N$. The product AB can be computed via a fast algorithm of order $\mathcal{O}(N \log N)$, with memory requirement of order $\mathcal{O}(N)$.*

Proof. Let $a = [a(1)\ a(2)\ \ldots\ a(N)]^T$ denote the first column of A and let $b = [b(1)\ b(2)\ \ldots\ b(N)]^T$ denote the first column of B. Furthermore, let $c = [c(1)\ c(2)\ \ldots\ c(N)]^T$ denote the first column of the product AB. Recall that the coordinates of the vector c are given by $c_k = \sum_{d|k, d\geq 1} a_d \cdot b_{k/d}$. Hence c is given by

$$\begin{bmatrix} c(1) \\ c(2) \\ c(3) \\ c(4) \\ c(5) \\ c(6) \\ \vdots \\ c(N) \end{bmatrix} = b(1)\begin{bmatrix} a(1) \\ a(2) \\ a(3) \\ a(4) \\ a(5) \\ a(6) \\ \vdots \\ a(N) \end{bmatrix} + b(2)\begin{bmatrix} \cdot \\ a(1) \\ \cdot \\ a(2) \\ \cdot \\ a(3) \\ \cdot \\ \vdots \end{bmatrix} + \cdots + b(N)\begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \\ \\ \cdot \\ \cdot \\ \cdot \\ a(1) \end{bmatrix}. \tag{3.6.6}$$

The above sum shows that c can be realized as $c = Ab$. (Equivalently, we could compute $c = Ba$.) Hence, the product AB of two D-matrices can be computed via Algorithm 3.6.2. $\square$

Proposition 3.6.8. *Let A be a D-matrix of size $N \times N$ and let v be an N- dimensional vector. Then $u = A^*v$ can be computed via a fast algorithm of order $\mathcal{O}(N \log N)$, with memory requirements of order $\mathcal{O}(N)$.*

Proof. Let $a = [a(1)\ a(2)\ \ldots\ a(N)]^T$ denote the first column of A. Recall that the matrix A^* is an upper triangular matrix with the k^{th} row featuring the first $\lfloor N/k \rfloor$ coefficients of $\overline{a}$ at each k^{th} position. The product $u = A^*v$ can be computed coordinate-wise via the following algorithm.

Algorithm 3.6.9. *For inputs a and v, initiate u as a zero vector of the same size as v via $u = zeros(size(v))$;.*

```
for k = 1 : N
    u(k) = conj(a(1 : floor(N/k))) * v(k : k : end);
end
```

The code above assumes that a is a row vector and v is a column vector. Alternatively, these conditions

may be lifted by checking if the inputs are of appropriate size and converting if necessary. This is done by including the following lines of code

$$
\begin{aligned}
&if\ iscolumn(a)\\
&\qquad a = a.';\\
&end\\
&if\ isrow(v)\\
&\qquad v = v.';\\
&end
\end{aligned}
$$

Observe that at each iteration of the loop, the number of operations is $2\lfloor N/k \rfloor - 1$. Hence the total number of operations is

$$\sum_{k=1}^{N} 2\lfloor N/k \rfloor - 1 = \mathcal{O}(N \log N). \tag{3.6.7}$$

□

Similarly to Algorithm 3.6.2, the preceding algorithm admits the following generalization.

Algorithm 3.6.10. *For input row vector a and matrix V, initiate U as a zero matrix of the same size as V via $U = zeros(size(V))$;.*

$$
\begin{aligned}
&for\ k = 1 : N\\
&\qquad U(k,:) = conj(a(1 : floor(N/k))) * V(k : k : end,:);\\
&end
\end{aligned}
$$

Observe that we can use Algorithms 3.6.2 and 3.6.9, in succession, to evaluate AA^* or A^*A at a vector v via fast algorithms, with memory requirements of order $\mathcal{O}(N)$.

3.7 Actions by Elementary D-matrices

In this section I discuss the actions of elementary D-matrices on the rows and columns of D-matrices. I will show that the actions in the infinite case are substantially different from those in the finite case.

3.7.1 Actions by Infinite Elementary D-matrices

Recall the elementary D-matrices D_2 and D_3:

$$D_2 = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \quad D_3 = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

Observe that the rows of an elementary D-matrix may be obtained by "stretching" or "spreading" the rows of the identity matrix I by a constant factor. If we consider D_2 and D_3 for example, we see that the first row of the identity matrix is featured as the second row in D_2, and as the third row in D_3. Similarly, the second row of I is featured as the fourth row of D_2, and as the sixth row of D_3. In general, if we consider D_n, we see that the k^{th} row of the identity matrix is the $(nk)^{\text{th}}$ row of D_n, with $n-1$ rows of zeros between rows nk and $n(k+1)$.

Furthermore, we see that the row action of D_n can be reversed by "sampling" every n^{th} row of D_n. This amounts to multiplication, on the left, by the transposed elementary D-matrix D_n^T. This shows, furthermore, that while D_n is not an invertible matrix for $n > 1$, we have that D_n^T is a left-inverse (outside of the Dirichlet ring) of D_n. Note also, that for $n > 1$ the elementary D-matrix D_n has infinitely many left-inverses.

Proposition 3.7.1. *Let D_n be an elementary D-matrix. If $n > 1$, then D_n has infinitely many left inverses. If L is a left-inverse of D_n, then for every $k \in \mathbb{N}$, the $(nk)^{th}$ column of L coincides with the $(nk)^{th}$ column of D_n^T. In particular, D_n^T is a left inverse of D_n.*

Proof. I have already shown that $D_n^T D_n = I$ by noting that the matrix D_n^T reverses the row-action of D_n. Alternatively, we could have shown the same by noting that the column action of D_n reverses the column action of D_n^T. Indeed, since the rows of D_n represent a "stretching" of the rows of the identity, we have that the columns of D_n^T represent a "stretching" of the columns of the identity. This shows that the action of D_n on the columns of D_n^T must represent "sampling" of every n^{th} column. Specifically, the k^{th} column of D_n is the $(nk)^{\text{th}}$ column of the identity matrix. This confirms the fact that the $(nk)^{\text{th}}$ column of D_n^T is the k^{th} column of the identity. Furthermore, this shows that any infinite matrix having the k^{th} column of the identity as its $(nk)^{\text{th}}$ column, is a left-inverse of D_n. □

Remark 3.7.2. *Since the non-zero rows of D_n coincide with the rows of the identity matrix, we have that D_n defines an isometric immersion of a Hilbert space onto a proper subset of itself. More precisely, if D_n acts on a Hilbert space $\mathbb{H}$, then for every $x = (x_1, x_2, \dots)^T \in \mathbb{H}$ we have that*

$$\|x\|^2 = \sum_{k=1}^{\infty} |x_k|^2 = \|D_n x\|^2. \tag{3.7.1}$$

As noted in the proof of the foregoing proposition, the columns of D_n represent a "sampling" of every n^{th} column of the identity matrix. This means that for $n > 1$, the column space of D_n is a proper subspace of the column space of the identity matrix. Consequently, we have that the column action of D_n, for $n > 2$, cannot be reversed, and hence it follows that D_n has no right-inverse. The action of "stretching" the columns of D_n by a factor of n is equivalent to right-multiplication by D_n^T. The action of multiplying by $D_n D_n^T$ on the right (or on the left) results in "stretching" the diagonal of the identity matrix by a factor of n, that is, the matrix $D_n D_n^T$ is the diagonal matrix with 1 in every kn^{th} position, for $k \in \mathbb{N}$, and zeros elsewhere. To visualize the right and left actions of D_n on D_n^T, we consider the case $n = 2$:

$$D_2^T D_2 = \begin{bmatrix} 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \quad D_2 D_2^T = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

Curiously, the commutator

$$[D_n^T, D_n] = D_n^T D_n - D_n D_n^T, \tag{3.7.2}$$

is the diagonal matrix with 1's in every k^{th} position, for $k \in \mathbb{N}$ satisfying $n \nmid k$, and zeros elsewhere. Hence, acting by $[D_n, D_n^T]$ on the left of a matrix A, represents sampling of every k^{th} row, for $n \nmid k$, while action on the right represents sampling of every k^{th} column, for $n \nmid k$. Applying the commutator from both sides results in sampling of every k^{th} row and every k^{th} column of A, for $n \nmid k$.

Observe that the actions of "stretching" the rows by a factor of n and the "sampling" of every n^{th} column are equivalent on all D-matrices. The reason for this lies in the commutativity of the Dirichlet ring. Furthermore, since for a D-matrix A, we have that $D_n A = A D_n$, we also have that

$$D_n^T A D_n = A \tag{3.7.3}$$

and

$$D_n^T A^* D_n = A^*. \tag{3.7.4}$$

3.7.2 Actions by Finite Elementary D-matrices

Significantly, the situation is very different when we consider actions by elementary D-matrices truncated to the upper left corner of size $N \times N$. My first observation is that, while in the infinite case, for every $n \in \mathbb{N}$, the corresponding elementary D-matrix D_n has a left inverse, in the finite case, all elementary D-matrices

D_n, for $n > 1$, are nilpotent. Next let us consider the products $D_n^T D_n$ and $D_n D_n^T$. In the infinite case, we established that acting by D_n^T on the left reversed the row action of D_n and hence we showed that $D_n^T D_n = I$. In the finite case, however, introducing a "stretching" factor of n to the rows of a matrix, results in losing all but the first $\lfloor N/n \rfloor$ rows. Therefore it is impossible to completely reverse the action. Instead, acting by $D_n^T D_n$ on the left, results in "sampling" of the first $\lfloor N/n \rfloor$ rows. Similarly, acting by $D_n^T D_n$ on the right results in "sampling" of the first $\lfloor N/n \rfloor$ columns. We have proven:

Proposition 3.7.3. *Let D_n be a finite elementary D-matrix of size $N \times N$ with $1 \leq n \leq N$. Then the matrix $D_n^T D_n$ is the diagonal projection matrix, with 1's in the first $\lfloor N/n \rfloor$ diagonal entries and 0's everywhere else. Consequently, given another finite elementary D-matrix D_m, we have that $D_m^T D_m = D_n^T D_n$ if and only if $\lfloor N/m \rfloor = \lfloor N/n \rfloor$.*

Corollary 3.7.4. *Let D_n be a finite elementary D-matrix of size $N \times N$ with $1 \leq n \leq N$. Define the matrix $I_n := D_n^T D_n - D_{n+1}^T D_{n+1}$. If $\lfloor N/(n+1) \rfloor \neq \lfloor N/n \rfloor$, then I_n is the diagonal projection matrix with 1's in every l^{th} diagonal position, for $\lfloor N/(n+1) \rfloor < l \leq \lfloor N/n \rfloor$, and 0's everywhere else.*

Recall that by Proposition 3.5.2, for any infinite D-matrix A, we have that the Hermitian matrix $A_n A_n^*$ is compatible with truncation. This holds, in particular, for elementary D-matrices. Hence, the matrix $D_n D_n^T$ is the matrix with 1's on the kn^{th} diagonal position, for every $k \leq \lfloor N/n \rfloor$, and zeros elsewhere. Below we have the matrices $D_2^T D_2$ and $D_2 D_2^T$ of size 8×8:

$$D_2^T D_2 = \begin{bmatrix} 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}, \quad D_2 D_2^T = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \end{bmatrix}. \tag{3.7.5}$$

In the finite case, the commutator

$$[D_n^T, D_n] = D_n^T D_n - D_n D_n^T \tag{3.7.6}$$

is again a diagonal matrix. However, since $D_n^T D_n$ has 1's in the first $\lfloor N/n \rfloor$ diagonal entries and 0's elsewhere, while $D_n D_n^T$ has 1's on the kn^{th} diagonal position, for every $k \leq \lfloor N/n \rfloor$, and 0's elsewhere, we have that $[D_n^T, D_n]$ has 1's in every l^{th} position, for $l \leq \lfloor N/n \rfloor$ satisfying $n \nmid l$, it has -1's in every kn^{th} diagonal position, for every k satisfying $\lfloor N/n \rfloor < kn \leq N$, and zeros elsewhere. Clearly the commutator is not compatible with truncation.

Recall that in the infinite case we established that $D_n^T A D_n = A$ and that $D_n^T A^* D_n = A^*$. While the same does not hold in the finite case, we have the following identities:

Proposition 3.7.5. *Let A be a finite D-matrix of size $N \times N$, and let D_n be an elementary D-matrix, with $1 \leq n \leq N$. Then we have that*

$$D_n^T A D_n = D_n^T (D_n A D_n^T) D_n. \tag{3.7.7}$$

Proof. Since $AD_n = D_n A$ we have that

$$D_n^T (D_n A D_n^T) D_n = (D_n^T A) D_n D_n^T D_n. \tag{3.7.8}$$

Hence, it suffices to show that $D_n D_n^T D_n = D_n$. Since the column action of D_n "samples" every n^{th} column, and since every n^{th} column of $D_n D_n^T$ coincides with the corresponding column of the identity matrix, we have that the matrix $D_n D_n^T D_n$ is the matrix obtained by "sampling" every n^{th} column of the identity matrix. But that is precisely D_n. □

Proposition 3.7.6. *Let A be a finite D-matrix of size $N \times N$, and let D_n be an elementary D-matrix, with $1 \leq n \leq N$. Then we have that*

$$D_n^T A^* D_n = D_n^T (D_n A^* D_n^T) D_n. \tag{3.7.9}$$

Proof. Since $A^* D_n^T = (D_n A)^* = (A D_n)^* = D_n^T A^*$, we have that

$$D_n^T (D_n A^* D_n^T) D_n = D_n^T D_n D_n^T (A^* D_n). \tag{3.7.10}$$

In the proof of the foregoing proposition we showed that $D_n D_n^T D_n = D_n$. Now the result follows from the fact that $D_n^T D_n D_n^T = (D_n D_n^T D_n)^T = D_n^T$. □

Chapter 4

Finite Band Decompositions

In this chapter I will look more closely into the number-theoretical structure of finite D-matrices, of size $N \times N$. I will begin the discussion by introducing a natural band decomposition of the set $\{1, 2, \ldots, N\}$ according to the distinct values that the floor function assumes on the set $\{N/1, N/2, \ldots, 1\}$. I will refer to this decomposition as the floor decomposition. Next, I will consider the partition of the columns of a D-matrix, induced by the floor decomposition. I will show that the columns in each band are mutually orthogonal and of equal norm, hence I will refer to this partition as the orthogonal band decomposition. I will also show how this decomposition induces a non-trivial resolution of the identity. Finally, I will show that the orthogonal band decomposition of a D-matrix induces a layer decomposition in associated Hermitian operators which can be applied to signal processing.

4.1 The Floor Band Decomposition

For a fixed positive number $N \in \mathbb{N}$, define the set $\mathcal{F}(N)$ to be the set of distinct positive floor values resulting from division of N by all positive integers bounded above by N, that is,

$$\mathcal{F}(N) := \{\lfloor N/m \rfloor \mid 1 \leq m \leq N\}. \tag{4.1.1}$$

For every floor value $k \in \mathcal{F}(N)$, define the set $\mathcal{F}_k(N)$ to be the set of all positive integers resulting in floor k when dividing N, that is,

$$\mathcal{F}_k(N) := \{m \in \mathbb{N} \mid \lfloor N/m \rfloor = k\}. \tag{4.1.2}$$

Observe that, for every $k \in \mathcal{F}(N)$, the elements of the set $\mathcal{F}_k(N)$ are bounded above by N. Furthermore, for every $n \in \{1, 2, \ldots, N\}$, we clearly have that $n \in \mathcal{F}_{\lfloor N/n \rfloor}(N)$. Since the sets $\mathcal{F}_k(N)$ and $\mathcal{F}_l(N)$ are disjoint, for $k \neq l \in \mathcal{F}(N)$, we have that the set $\mathcal{F}(N)$ partitions the set $\{1, 2, \ldots, N\}$ into the classes $\mathcal{F}_k(N)$, that is,

$$\{1, 2, \ldots, N\} = \bigsqcup_{k \in \mathcal{F}(N)} \mathcal{F}_k(N). \tag{4.1.3}$$

Since the sets $\{\mathcal{F}_k(N) \mid k \in \mathcal{F}(N)\}$ partition the set $\{1, 2, \ldots, N\}$ into bands, we will refer to this partition as the **floor band decomposition** of N. Furthermore, we will refer to the sets $\mathcal{F}_k(N)$ as the **floor bands** and we will refer to the set $\mathcal{F}(N)$ as the **set of floors.** For brevity, whenever N is fixed, we will write $\mathcal{F}$ and $\mathcal{F}_k$ instead of $\mathcal{F}(N)$ and $\mathcal{F}_k(N)$, respectively.

Example 4.1.1. *For $N = 8$, the floor set is $\mathcal{F} = \{8, 4, 2, 1\}$ and the corresponding floor bands are $\mathcal{F}_8 = \{1\}$, $\mathcal{F}_4 = \{2\}$, $\mathcal{F}_2 = \{3, 4\}$ and $\mathcal{F}_1 = \{5, 6, 7, 8\}$.*

Note that for $N = 8$ the set of floors coincides with the set of distinct positive divisors of N. In general, however, we cannot expect the floor and divisor sets to coincide. To demonstrate this, we consider an example with a prime N.

Example 4.1.2. *For $N = 7$, the set of floors is $\mathcal{F} = \{7, 3, 2, 1\}$ and the corresponding floor bands are $\mathcal{F}_7 = \{1\}$, $\mathcal{F}_3 = \{2\}$, $\mathcal{F}_2 = \{3\}$ and $\mathcal{F}_1 = \{4, 5, 6, 7\}$.*

Lemma 4.1.3. *Let N be a positive integer. Take $m < l \in \mathcal{F}_k$, for some $k \in \mathcal{F}$. Then we have that $lcm(m, l) > N$.*

Proof. Let $L = lcm(m, l)$ and assume that $L \leq N$. Since $m, l \mid L$, we have that

$$\frac{L}{m} + \left\lfloor \frac{N-L}{m} \right\rfloor = \left\lfloor \frac{N}{m} \right\rfloor = \left\lfloor \frac{N}{l} \right\rfloor = \frac{L}{l} + \left\lfloor \frac{N-L}{l} \right\rfloor. \tag{4.1.4}$$

Since $m < l$ we have that

$$\frac{L}{m} > \frac{L}{l} \tag{4.1.5}$$

which implies that

$$\left\lfloor \frac{N-L}{m} \right\rfloor < \left\lfloor \frac{N-L}{l} \right\rfloor. \tag{4.1.6}$$

However, this implies that $m > l$, which is a contradiction to our choice of m and l. Therefore, it follows that L must be greater than N. □

Lemma 4.1.4. *Let N be a positive integer. For every $k \in \mathcal{F}$ we have that*

$$\mathcal{F}_k = \{m \in \mathbb{N} \mid \lfloor N/(k+1) \rfloor < m \leq \lfloor N/k \rfloor\}. \tag{4.1.7}$$

Proof. By definition, $m \in \mathcal{F}_k$ if and only if $\lfloor N/m \rfloor = k$. Equivalently, $k \leq N/m < k + 1$. Rearranging this yields $\frac{N}{k+1} < m \leq \frac{N}{k}$. Since m is an integer, we get that $\frac{N}{k+1} < m \leq \lfloor \frac{N}{k} \rfloor$.

□

4.2 The Orthogonal Band Decomposition

In this section I will discuss the natural non-trivial decomposition of a finite D-matrix A of size $N \times N$, induced by the floor band decomposition of N. I will show that the floor band decomposition of N partitions A into orthogonal bands A_k. Furthermore, I will show that for every invertible D-matrix the orthogonal band decomposition defines a non-trivial resolution of the identity matrix.

I begin the discussion by introducing the partition of the identity matrix induced by the floor band decomposition.

Lemma 4.2.1. *Let N be a positive integer. Then the identity matrix I of size $N \times N$ admits the following band decomposition*

$$I = \sum_{k \in \mathcal{F}} I_k, \tag{4.2.1}$$

where $I_k = D_k^T D_k - D_{k+1}^T D_{k+1}$.

Proof. Take $k \in \mathcal{F}$. Recall that, by Lemma 4.1.4, we have that $\mathcal{F}_k = \{m \in \mathbb{N} \mid \lfloor N/(k+1) \rfloor < m \leq \lfloor N/k \rfloor\}$. This, together with Corollary 3.7.4, imply that I_k is the diagonal matrix with 1's located in the diagonal positions indexed by the floor band $\mathcal{F}_k$. In other words, I_k is the matrix whose non-zero columns feature the columns of the identity matrix indexed by the floor band $\mathcal{F}_k$. Since the set of floor bands partitions the set $\{1, 2, \ldots, N\}$, we have that the set $\{I_k \mid k \in \mathcal{F}\}$ decomposes the identity matrix into consecutive bands. □

Theorem 4.2.2. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite invertible D-matrix of size $N \times N$. For every $k \in \mathcal{F}$, let $A_k = AI_k$. Then we have that*

$$A = \sum_{k \in \mathcal{F}} A_k. \tag{4.2.2}$$

Furthermore, for every $k \in \mathcal{F}$ we have that

$$A_k^* A_k = \lambda_k I_k, \tag{4.2.3}$$

where $\lambda_k = \sum_{n=1}^{k} |a_n|^2$. Consequently, the identity matrix admits the following resolution

$$I = \sum_{k \in \mathcal{F}} \lambda_k^{-1} A_k^* A_k. \tag{4.2.4}$$

Proof. Note that, by definition, the non-zero columns of the matrix A_k correspond to the non-zero columns of the matrix I_k. Since the non-zero columns of I_k are indexed by the floor band $\mathcal{F}_k$, we have that the matrix A_k is obtained by selecting the columns of A indexed by $\mathcal{F}_k$ and by setting all other columns to zero. Hence we have that the set $\{A_k \mid k \in \mathcal{F}\}$ is a band decomposition of A.

Now take $m \in \mathcal{F}_k$. Recall that, by Proposition 3.4.3, the m^{th} column of A features the first $\lfloor N/m \rfloor$ elements of the generating sequence. Since the set $\mathcal{F}_k$ consists of all m such that $\lfloor N/m \rfloor = k$, it follows that the matrix A_k is the matrix whose non-zero columns are the columns of A featuring the first k elements of the generating sequence (a_n). From here we have that the non-zero columns of A_k have the same norm, equal to $\sqrt{\lambda_k}$.

Next, we need to show that the columns of A_k are mutually orthogonal. If A_k has only one non-zero column, then there is nothing to show. Suppose that A_k has at least two non-zero columns and let m and n denote the indices of two such columns. Since $m, n \in \mathcal{F}_k$, by Lemma 4.1.3, we have that $\operatorname{lcm}(m, n) > N$. By Corollary 3.5.5, this implies that the m^{th} and n^{th} columns of the matrix A_k are mutually orthogonal.

Finally, by Lemma 4.2.1 we have that $I = \sum_{k \in \mathcal{F}} \lambda_k^{-1} A_k^* A_k$. □

Corollary 4.2.3. *Let $A = \sum_{n=1}^{N} a_n D_n$ be an invertible finite D-matrix of size $N \times N$. For each $k \in \mathcal{F}$ we have that the operator norm of A_k is equal to $\sqrt{\lambda_k}$, i.e.,*

$$\|A_k\| = \sqrt{\lambda_k}. \tag{4.2.5}$$

Proof. By Theorem 4.2.2, we have that $A_k^* A_k = \lambda_k I_k$. Hence $A_k^* A_k$ has exactly two eigen values, namely λ_k and 0. Since A is invertible, we have that $\lambda_k > 0$. This implies that $\|A_k^* A_k\| = \lambda_k$. Consequently, we have that $\|A_k\| = \sqrt{\lambda_k}$. □

Corollary 4.2.4. *Let $A = \sum_{n=1}^{N} a_n D_n$ be an invertible finite D-matrix of size $N \times N$. Then the norms of the orthogonal bands of A provide bounds for the operator norm of A, i.e.,*

$$\sqrt{\lambda_N} \leq \|A\| \leq \sum_{k \in \mathcal{F}} \sqrt{\lambda_k}. \tag{4.2.6}$$

Proof. By definition, $A_k = AI_k$ for every $k \in \mathcal{F}$. Hence, by Corollary 4.2.3 we have that

$$\sqrt{\lambda_k} = \|A_k\| \leq \|A\| \cdot \|I_k\| = \|A\|, \tag{4.2.7}$$

for every $k \in \mathcal{F}$. In particular $\sqrt{\lambda_N} \leq \|A\|$. The upper bound for the operator norm of A follows by a direct application of the triangle inequality on the sum $\sum_{k \in \mathcal{F}} A_k = A$. □

Corollary 4.2.5. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite invertible D-matrix of size $N \times N$. Then we have that the matrix*

$$\Lambda := \sum_{k \in \mathcal{F}} A_k^* A_k \tag{4.2.8}$$

is a diagonal matrix. The partition induces by the floor band decomposition of N is $\{\lambda_k I_k \mid k \in \mathcal{F}\}$, that is,

$$\Lambda = \sum_{k \in \mathcal{F}} \lambda_k I_k. \tag{4.2.9}$$

Henceforth, given a D-matrix $A = \sum_{n=1}^{N} a_n D_n$, I will refer to the set $\{A_k \mid k \in \mathcal{F}\}$ as the **orthogonal band decomposition** of A and I will refer to the matrices A_k as the **orthogonal bands** of A. I will fix the notation λ_k to denote the square of the norm of the columns of A indexed by the floor band $\mathcal{F}_k$. Finally, I will refer to the diagonal matrix Λ as the **norm matrix** associated with A.

Example 4.2.6. *Consider the following D-matrix A with $N = 8$:*

$$A = \begin{bmatrix} 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 6 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 2 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 8 & 6 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\ 5 & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\ 3 & 2 & 6 & \cdot & \cdot & 1 & \cdot & \cdot \\ 7 & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\ 4 & 8 & \cdot & 6 & \cdot & \cdot & \cdot & 1 \end{bmatrix}.$$

The matrix A decomposes into 4 orthogonal bands, A_1, A_2, A_4 and A_8.

$$\lambda_8 = 204, \quad A_8 = \begin{bmatrix} 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 6 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 2 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 8 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 5 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 3 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 7 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 4 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}, \quad \lambda_4 = 105, \quad A_4 = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 6 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 2 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 8 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix},$$

$$\lambda_2 = 37, \quad A_2 = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 6 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 6 & \cdot & \cdot & \cdot & \cdot \end{bmatrix}, \quad \lambda_1 = 1, \; A_1 = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \end{bmatrix}.$$

4.3 The Band Decomposition of AA^*

Proposition 4.3.1. *Let $A = \sum_{n=1}^{N} a_n D_n$ be an invertible finite D-matrix of size $N \times N$. Then for every $k \in \mathcal{F}$ we have that*

$$A_k = \sum_{m \in \mathcal{F}_k} D_m A_N D_m^T, \tag{4.3.1}$$

where the m^{th} column of $D_m A_N D_m^T$ is the m^{th} column of A and all other columns of $D_m A_N D_m^T$ are zero.

Proof. Recall that A_N features the first column of A as its first column and has all other columns equal to zero. Note also that $\mathcal{F}_N = \{1\}$. For $m < N$, the matrix $D_m A_N$ is obtained by "spreading" the rows of A_N by a factor of m. Hence the matrix $D_m A_N$ features the first $\lfloor N/m \rfloor$ terms of the generating sequence in its first column, spread by a factor of m, and has all other columns equal to zero. Now the matrix $D_m A_N D_m^T$ is obtained by "spreading" the columns of $D_m A_N$ by a factor of m. Since the only non-zero column of $D_m A_N$ is the first column, we have that $D_m A_N D_m^T$ is obtained by placing the first column of $D_m A_N$ in the m^{th} column and setting all other columns equal to zero. But that is the same as selecting the m^{th} column of A and setting all other columns to zero. Now the result follows from the fact that, for every $k \in \mathcal{F}$, the matrix A_k is obtained by selecting the columns of A featuring the first k terms of the generating sequence and setting all other columns to zero. □

Lemma 4.3.2. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite invertible D-matrix of size $N \times N$. Then for every $k \in \mathcal{F}$ we have that*

$$A_k A_k^* = \sum_{m \in \mathcal{F}_k} D_m (A_N A_N^*) D_m^T \tag{4.3.2}$$

Proof. Take $k \in \mathcal{F}$ and $l, m \in \mathcal{F}_k$, with $l \neq m$. By Lemma 4.1.3, we have that $\operatorname{lcm}(l, m) > N$. By Proposition 3.5.4 it follows that $D_m^T D_l = 0$. From here we have that

$$\begin{aligned}
A_k A_k^* &= \sum_{m \in \mathcal{F}_k} D_m A_N D_m^T \cdot \sum_{l \in \mathcal{F}_k} D_l A_N^* D_l^T && (4.3.3)\\
&= \sum_{m \in \mathcal{F}_k} \sum_{l \in \mathcal{F}_k} D_m A_N D_m^T \cdot D_l A_N^* D_l^T && (4.3.4)\\
&= \sum_{m \in \mathcal{F}_k} D_m A_N (D_m^T D_m) A_N^* D_m^T && (4.3.5)\\
&= \sum_{m \in \mathcal{F}_k} A_N (D_m D_m^T)^2 A_N^* && (4.3.6)\\
&= \sum_{m \in \mathcal{F}_k} A_N (D_m D_m^T) A_N^* && (4.3.7)\\
&= \sum_{m \in \mathcal{F}_k} D_m (A_N A_N^*) D_m^T. && (4.3.8)
\end{aligned}$$

□

The following example provides a visualization of Lemma 4.3.2.

Example 4.3.3. *Consider the matrix A of Example 4.2.6. The matrices $A_4 A_4^*, A_2 A_2^*$ and $A_1 A_1^*$ exhibit the matrix $A_8 A_8^*$ truncated to the upper left corners of size $4 \times 4, 2 \times 2$ and 1×1, respectively.*

$$A_8 A_8^* = \begin{bmatrix}
1 & 6 & 2 & 8 & 5 & 3 & 7 & 4 \\
6 & 36 & 12 & 48 & 30 & 18 & 42 & 24 \\
2 & 12 & 4 & 16 & 10 & 6 & 14 & 8 \\
8 & 48 & 12 & 64 & 40 & 24 & 56 & 32 \\
5 & 30 & 10 & 40 & 25 & 15 & 35 & 20 \\
3 & 18 & 6 & 24 & 15 & 9 & 21 & 12 \\
7 & 42 & 14 & 56 & 35 & 21 & 49 & 28 \\
4 & 24 & 8 & 32 & 20 & 12 & 28 & 16
\end{bmatrix}, \quad A_4 A_4^* = \begin{bmatrix}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & 1 & \cdot & 6 & \cdot & 2 & \cdot & 8 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & 6 & \cdot & 36 & \cdot & 12 & \cdot & 48 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & 2 & \cdot & 12 & \cdot & 4 & \cdot & 16 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & 8 & \cdot & 48 & \cdot & 16 & \cdot & 64
\end{bmatrix},$$

$$A_2A_2^* = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot & 6 & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & 6 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 6 & \cdot & \cdot & 36 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 6 & \cdot & \cdot & \cdot & 36 \end{bmatrix}, \quad A_1A_1^* = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \end{bmatrix}.$$

Theorem 4.3.4. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite invertible D-matrix of size $N \times N$. The floor band decomposition of N induces the following decomposition of the Hermitian matrix AA^**

$$AA^* = \sum_{k \in \mathcal{F}} A_k A_k^*. \tag{4.3.9}$$

Proof. Recall that $A_k = AI_k$. Hence we have that

$$A_k A_k^* = AI_k I_k A^* = AI_k A^*, \tag{4.3.10}$$

where the latter equality holds since I_k is a projection matrix. From here, together with Lemma 4.2.1, we have that

$$\sum_{k \in \mathcal{F}} A_k A_k^* = \sum_{k \in \mathcal{F}} AI_k A^* = A \left(\sum_{k \in \mathcal{F}} I_k \right) A^* = AA^*. \tag{4.3.11}$$

□

Corollary 4.3.5. *Let $A = \sum_{n=1}^{N} a_n D_n$ be an invertible finite D-matrix of size $N \times N$. Then we have that*

$$A^*A = A^{-1} \left(\sum_{k \in \mathcal{F}} A_k A_k^* \right) A. \tag{4.3.12}$$

Corollary 4.3.6. *Let $A = \sum_{n=1}^{N} a_n D_n$ be an invertible finite D-matrix of size $N \times N$. Then we have that*

$$[A_k, A_k^*] = AI_k A^* - \lambda_k I_k \tag{4.3.13}$$

and

$$\sum_{k \in \mathcal{F}} [A_k, A_k^*] = AA^* - \Lambda. \tag{4.3.14}$$

4.4 The Associated P-matrix and its Band Decomposition

I will next consider the orthogonal projections P_k from the column space of A onto the subspaces generated by each orthogonal band A_k of A. I will refer to the matrices P_k as the **band projections** of A.

Lemma 4.4.1. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite invertible D-matrix of size $N \times N$ and let A_k be an orthogonal band of A. The corresponding projection matrix P_k, onto the column space of A_k, is given by*

$$P_k = \lambda_k^{-1} A_k A_k^*. \tag{4.4.1}$$

Furthermore, for every $k \in \mathcal{F}$ we have that the band projection P_k is a norm 1 operator, that is,

$$\|P_k\| = 1. \tag{4.4.2}$$

Proof. Let C_k be the $N \times (\lfloor N/k \rfloor - \lfloor N/(k+1) \rfloor)$ matrix consisting of the non-zero columns of A_k. The projection matrix onto the column space of C_k (or equivalently A_k) is given by

$$P_k = C_k (C_k^* C_k)^{-1} C_k^*. \tag{4.4.3}$$

Since the non-zero columns of A_k are mutually orthogonal, and each non-zero column features the first k terms of the generating sequence (a_n), we have that the matrix $C_k^* C_k$ is a diagonal matrix with λ_k along the diagonal. From here it follows that

$$P_k = \lambda_k^{-1} C_k C_k^*. \tag{4.4.4}$$

Now, observe that each row of A_k has the same non-zero elements as the corresponding row in C_k. Since each coordinate of $C_k C_k^*$ is the inner product of two rows of C_k, and therefore the inner product of the corresponding rows of A_k, it follows that

$$C_k C_k^* = A_k A_k^*. \tag{4.4.5}$$

Finally, since $\|A_k A_k^*\| = \|A_k\|^2$, by Proposition 4.2.3, we have that

$$\|P_k\| = \lambda_k^{-1} \|A_k A_k^*\| = \lambda_k^{-1} \lambda_k = 1. \tag{4.4.6}$$

□

Corollary 4.4.2. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite invertible D-matrix of size $N \times N$. Then every band projection is Hermitian, that is, $P_k = P_k^*$.*

Theorem 4.4.3. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite invertible D-matrix of size $N \times N$. Define $P := \sum_{k \in \mathcal{F}} P_k$ to be the sum of the band projections of A. Then P admits the following decomposition*

$$P = A \Lambda^{-1} A^*. \tag{4.4.7}$$

Furthermore, we have that

$$\|P\| = \frac{1}{a_1^2} \|A\|^2. \tag{4.4.8}$$

Proof. Since by definition, $P_k = \lambda_k^{-1} A_k A_k^*$ and $A_k = AI_k$, we have that

$$P = \sum_{k\in\mathcal{F}} \lambda_k^{-1} A_k A_k^* \tag{4.4.9}$$

$$= \sum_{k\in\mathcal{F}} \lambda_k^{-1} AI_k A^* \tag{4.4.10}$$

$$= A\left(\sum_{k\in\mathcal{F}} \lambda_k^{-1} I_k\right) A^* \tag{4.4.11}$$

$$= A\Lambda^{-1}A^*. \tag{4.4.12}$$

From here, it follows that

$$\|P\| \leq \|A\| \cdot \|\Lambda^{-1}\| \cdot \|A^*\|. \tag{4.4.13}$$

Since $\|A\| = \|A^*\|$ and $\|\Lambda^{-1}\| = \max_k\{\lambda_k^{-1}\} = 1/a_1$, we have that $\|P\| \leq \frac{1}{a_1^2}\|A\|^2$. On the other hand, we have that

$$\frac{1}{a_1^2} = \|\Lambda^{-1}\| = \|A^{-1}P(A^{-1})^*\| \tag{4.4.14}$$

$$\leq \|A^{-1}\| \cdot \|P\| \cdot \|(A^{-1})^*\| \tag{4.4.15}$$

$$= \frac{\|P\|}{\|A\|^2}. \tag{4.4.16}$$

Hence, we have that

$$\|P\| = \frac{1}{a_1^2}\|A\|^2. \tag{4.4.17}$$

□

We will refer to the matrix $P = A\Lambda^{-1}A^*$ as the P-matrix associated with A, or the **P-matrix** of A.

Corollary 4.4.4. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite invertible D-matrix of size $N \times N$. Then we have that $P^{-1} = (A^{-1})^*\Lambda A^{-1}$.*

Corollary 4.4.5. *Let A be a finite invertible D-matrix. Then the P-matrix of A is a Hermitian invertible matrix and the same is true for its inverse, i.e., $P = P^*$ and $P^{-1} = (P^{-1})^*$.*

Theorem 4.4.6. *Let A be a finite invertible D-matrix of size $N \times N$ and let P be its P-matrix. Furthermore, for every $k \in \mathcal{F}$, let $B_k = I_k A^{-1}$ be the matrix whose non-zero rows are the rows of A^{-1} indexed by the band $\mathcal{F}_k$. Then the inverse of P admits the following band decomposition*

$$P^{-1} = \sum_{k\in\mathcal{F}} \lambda_k B_k^* B_k. \tag{4.4.18}$$

Proof. From Corollary 4.4.4 we have that

$$P^{-1} = (A^{-1})^*\Lambda A^{-1}. \tag{4.4.19}$$

Since $\Lambda = \sum_{k\in\mathcal{F}} \lambda_k I_k$, and the matrices I_k are real and idempotent, we have that

$$P^{-1} = \sum_{k\in\mathcal{F}} \lambda_k (A^{-1})^* I_k A^{-1} \tag{4.4.20}$$

$$= \sum_{k\in\mathcal{F}} \lambda_k (I_k A^{-1})^* (I_k A^{-1}) \tag{4.4.21}$$

$$= \sum_{k\in\mathcal{F}} \lambda_k B_k^* B_k. \tag{4.4.22}$$

□

Example 4.4.7. *Consider the matrix A of Example 4.2.6. The associated P-matrix is*

$$P = \begin{bmatrix} 0.0049 & 0.0294 & 0.0098 & 0.0392 & 0.0245 & 0.0147 & 0.0343 & 0.0196 \\ 0.0294 & 0.1860 & 0.0588 & 0.2924 & 0.1471 & 0.1073 & 0.2059 & 0.1938 \\ 0.0098 & 0.0588 & 0.0466 & 0.0784 & 0.0490 & 0.1916 & 0.0686 & 0.0392 \\ 0.0392 & 0.2924 & 0.0784 & 0.6836 & 0.1961 & 0.2319 & 0.2745 & 0.7762 \\ 0.0245 & 0.1471 & 0.0490 & 0.1961 & 1.1225 & 0.0735 & 0.1716 & 0.0980 \\ 0.0147 & 0.1073 & 0.1916 & 0.2319 & 0.0735 & 2.0552 & 0.1029 & 0.2112 \\ 0.0343 & 0.2059 & 0.0686 & 0.2745 & 0.1716 & 0.1029 & 1.2402 & 0.1373 \\ 0.0196 & 0.1938 & 0.0392 & 0.7762 & 0.0980 & 0.2112 & 0.1373 & 2.6609 \end{bmatrix}, \tag{4.4.23}$$

and its inverse is

$$P^{-1} = \begin{bmatrix} 49031 & -10360 & -200 & 1780 & -5 & 21 & -7 & -124 \\ -10360 & 2225 & 12 & -390 & \cdot & -2 & \cdot & 28 \\ -200 & 12 & 73 & \cdot & \cdot & -6 & \cdot & \cdot \\ 1780 & -390 & \cdot & 73 & \cdot & \cdot & \cdot & -6 \\ -5 & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\ 21 & -2 & -6 & \cdot & \cdot & 1 & \cdot & \cdot \\ -7 & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\ -124 & 28 & \cdot & -6 & \cdot & \cdot & \cdot & 1 \end{bmatrix}. \tag{4.4.24}$$

Note that the projection bands P_k have the same structure as the bands $A_k A_k^*$ of AA^*. Hence the projection bands satisfy the following lemma, which is analogous to Lemma 4.4.1.

Lemma 4.4.8. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite invertible D-matrix of size $N \times N$. Then for every $k \in \mathcal{F}$ we have that*

$$P_k = \lambda_N \lambda_k^{-1} \sum_{l\in\mathcal{F}_k} D_l P_N D_l^T. \tag{4.4.25}$$

We are now ready to prove the following result:

Theorem 4.4.9. *Let $A = \sum_{n=1}^{N} a_n D_n$ and $A' = \sum_{n=1}^{N} a'_n D_n$ be finite invertible D-matrices of size $N \times N$. Let P and P' denote the P-matrices of A and A', respectively. Then we have that*

$$P' = P \quad \text{if and only if} \quad A' = \alpha A, \tag{4.4.26}$$

for some $\alpha \in \mathbb{C}\backslash\{0\}$.

Proof. Suppose that $A' = \alpha A$, for some $\alpha \in \mathbb{C}\backslash\{0\}$. Then for every $k \in \mathcal{F}$, we have that $\lambda'_k = \alpha^2 \lambda_k$. This implies that $\Lambda' = \alpha^2 \Lambda$. Now, by Theorem 4.4.3 we have that

$$P' = A'(\Lambda')^{-1}(A')^* = \alpha A \alpha^{-2} \Lambda^{-1} \alpha A^* = A \Lambda^{-1} A^* = P. \tag{4.4.27}$$

Now suppose that $P' = P$. By Lemma 4.4.8, it follows that $P'_N = P_N$. Since $P_N = \lambda_N^{-1} A_N A_N^*$, we have that the first column of P_N is a multiple of the generating sequence (a_n) of A. More precisely, the first column of P_N is equal to $\lambda_N^{-1} \overline{a}_1 a$, where $a = [a_1, a_2, \ldots, a_N]^T$. Similarly, the first column of P'_N is equal to $(\lambda'_N)^{-1} \overline{a'}_1 a'$, where $a' = [a'_1, a'_2, \ldots, a'_N]^T$. From here we have that

$$(\lambda'_N)^{-1} \overline{a'}_1 a' = \lambda_N^{-1} \overline{a}_1 a, \tag{4.4.28}$$

or equivalently

$$a' = \frac{\lambda'_N \overline{a'}_1}{\lambda_N \overline{a}_1} a. \tag{4.4.29}$$

The result now follows from the fact that any D-matrix is uniquely determined by its generating sequence. □

4.5 Partial Reconstructions

Throughout this section I will consider a subset $\widetilde{\mathcal{F}}$ of $\mathcal{F}$ and a finite invertible D-matrix A of size $N \times N$. I will denote by $\widetilde{I}$, $\widetilde{A}$, $\widetilde{\Lambda}$, $\widetilde{P}$ and $\widetilde{P^{-1}}$ the sums of the respective band components indexed by $\widetilde{\mathcal{F}}$, i.e.,

$$\widetilde{I} = \sum_{k \in \widetilde{\mathcal{F}}} I_k; \quad \widetilde{A} = \sum_{k \in \widetilde{\mathcal{F}}} A_k; \quad \widetilde{\Lambda} = \sum_{k \in \widetilde{\mathcal{F}}} \lambda_k I_k; \quad \widetilde{P} = \sum_{k \in \widetilde{\mathcal{F}}} P_k; \quad \widetilde{P^{-1}} = \sum_{k \in \widetilde{\mathcal{F}}} \lambda_k B_k^* B_k. \tag{4.5.1}$$

Proposition 4.5.1. *Let* $A = \sum_{n=1}^{N} a_n D_n$ *be a finite invertible D-matrix of size* $N \times N$ *and let* $\widetilde{\mathcal{F}} \subset \mathcal{F}$. *Then we have that*

$$\widetilde{A} = A\widetilde{I}, \tag{4.5.2}$$

and consequently,

$$A^{-1}\widetilde{A} = \widetilde{I}. \tag{4.5.3}$$

Proof. Since $A_k = AI_k$, we have that $A = \sum_{k \in \widetilde{\mathcal{F}}} AI_k = A \sum_{k \in \widetilde{\mathcal{F}}} I_k = A\widetilde{I}$. □

Lemma 4.5.2. *Let* $A = \sum_{n=1}^{N} a_n D_n$ *be a finite invertible D-matrix of size* $N \times N$ *and let* $\widetilde{\mathcal{F}} \subset \mathcal{F}$. *Then we have that*

$$\widetilde{\Lambda} = \Lambda \widetilde{I}. \tag{4.5.4}$$

Proof. Consider the product $\Lambda\widetilde{I}$. We have that

$$\Lambda\widetilde{I} = \left(\sum_{l\in\mathcal{F}} \lambda_l I_l\right)\left(\sum_{k\in\widetilde{\mathcal{F}}} I_k\right) \tag{4.5.5}$$

$$= \sum_{l\in\mathcal{F}}\sum_{k\in\widetilde{\mathcal{F}}} \lambda_l (I_l I_k) \tag{4.5.6}$$

$$= \sum_{k\in\widetilde{\mathcal{F}}} \lambda_k I_k, \tag{4.5.7}$$

where the last equality follows from the fact that the columns of the identity matrix are mutually orthogonal and hence for $l \neq k \in \mathcal{F}$ it follows that $I_l I_k$ is the zero matrix. □

Corollary 4.5.3. *Let $M = \sum_{k\in\mathcal{F}} \mu_k I_k$ for $\mu_k \in \mathbb{C}$. Take $\widetilde{\mathcal{F}} \subset \mathcal{F}$ and let $\widetilde{M} = \sum_{k\in\widetilde{\mathcal{F}}} \mu_k I_k$. Then we have that*

$$\widetilde{M} = M\widetilde{I}. \tag{4.5.8}$$

Lemma 4.5.4. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite invertible D-matrix of size $N \times N$ and let $\widetilde{\mathcal{F}} \subset \mathcal{F}$. Then*

$$\widetilde{P} = A\widetilde{\Lambda^{-1}}A^*, \tag{4.5.9}$$

and

$$\widetilde{P^{-1}} = (A^{-1})^*\widetilde{\Lambda}A^{-1}. \tag{4.5.10}$$

Proof. From the proof of Theorem 4.4.3 we can see that

$$\widetilde{P} = \sum_{k\in\widetilde{\mathcal{F}}} P_k = A\left(\sum_{k\in\widetilde{\mathcal{F}}} \lambda_k^{-1} I_k\right)A^* = A\widetilde{\Lambda^{-1}}A^*. \tag{4.5.11}$$

Similarly, from the proof of Theorem 4.4.6, we have that

$$\widetilde{P^{-1}} = (A^{-1})^*\left(\sum_{k\in\widetilde{\mathcal{F}}} \lambda_k I_k\right)A^{-1} = (A^{-1})^*\widetilde{\Lambda}A^{-1}. \tag{4.5.12}$$

□

Corollary 4.5.5. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite invertible D-matrix of size $N \times N$ and let $\widetilde{\mathcal{F}} \subset \mathcal{F}$. Then we have that $\widetilde{P} = \widetilde{P}^*$ and $\widetilde{P^{-1}} = \widetilde{P^{-1}}^*$.*

Corollary 4.5.6. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite invertible D-matrix of size $N \times N$. Let $M = \sum_{k\in\mathcal{F}} \mu_k I_k$ for $\mu_k \in \mathbb{R}\backslash\{0\}$. For $T = AMA^*$ we have that $\widetilde{T} = A\widetilde{M}A^*$ and $\widetilde{T}^* = \widetilde{T}$.*

Theorem 4.5.7. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite invertible D-matrix of size $N \times N$ and let $\widetilde{\mathcal{F}} \subset \mathcal{F}$. Then we have that*

$$P^{-1}\widetilde{P} = \widetilde{P^{-1}}\widetilde{P}. \tag{4.5.13}$$

Proof. Using Lemma 4.5.4 and Corollary 4.5.3 we have that

$$P^{-1}\widetilde{P} = \left((A^*)^{-1}\Lambda A^{-1}\right)\left(A\widetilde{\Lambda}^{-1}A^*\right) \tag{4.5.14}$$

$$= (A^*)^{-1}\Lambda\Lambda^{-1}\widetilde{I}A^* \tag{4.5.15}$$

$$= (A^*)^{-1}\widetilde{I}A^*. \tag{4.5.16}$$

Since diagonal matrices commute, and since $\widetilde{I}$ is idempotent, we also have that

$$\widetilde{P^{-1}}\widetilde{P} = \left((A^*)^{-1}\widetilde{\Lambda}A^{-1}\right)\left(A\widetilde{\Lambda}^{-1}A^*\right) \tag{4.5.17}$$

$$= (A^*)^{-1}\widetilde{I}\Lambda\Lambda^{-1}\widetilde{I}A^* \tag{4.5.18}$$

$$= (A^*)^{-1}\widetilde{I}A^*. \tag{4.5.19}$$

□

Corollary 4.5.8. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite invertible D-matrix of size $N \times N$ and let $\widetilde{\mathcal{F}} \subset \mathcal{F}$. Then we have that the matrix $P^{-1}\widetilde{P}$ is idempotent, that is,*

$$(P^{-1}\widetilde{P})^2 = P^{-1}\widetilde{P}. \tag{4.5.20}$$

Furthermore, we have that

$$\widetilde{P}P^{-1} = A\widetilde{I}A^{-1} = (P^{-1}\widetilde{P})^*, \tag{4.5.21}$$

and hence $\widetilde{P}P^{-1}$ is idempotent as well.

Corollary 4.5.9. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite invertible D-matrix of size $N \times N$. Let $M = \sum_{k\in\mathcal{F}} \mu_k I_k$ for $\mu_k \in \mathbb{R}\backslash\{0\}$. For $T = AMA^*$ we have that $T^{-1}\widetilde{T} = P^{-1}\widetilde{P}$ and $\widetilde{T}T^{-1} = \widetilde{P}P^{-1}$.*

4.6 Computations with P and $\widetilde{P}$

Even though P-matrices are not sparse (see Example 4.4.7), they admit a fast implementation. In this section I will show how.

Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite D-matrix. The set of floors $\mathcal{F}$, the floor bands $\mathcal{F}_k$, and the distinct terms λ_k of the norm matrix Λ, can all be computed via the following algorithm of order $\mathcal{O}(N)$, with memory requirement of order $\mathcal{O}(N)$.

Algorithm 4.6.1. *For input vector a, we compute F, F_k and λ via*

$N = length(a);$
$K = 1 : N;$
$K = floor(N./K);$
$F = unique(K,\ 'stable');$ *%'stable' preserves the order.*
$L = length(F);$

$F_k = cell(1, L);$

$\lambda = zeros(1, L);$

$for\ k = 1 : L$

$\quad F_k(k) = find(K == F(k));$

$\quad \lambda(k) = (norm(a(1 : F(k)))).^2;$

end

Proposition 4.6.2. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite D-matrix and let P be the P-matrix associated with A. Let $v = [v(1)\ v(2)\ \dots\ v(N)]^T$ be a column vector of complex numbers. Then the product $u = Pv$ can be evaluated via a fast algorithm of order $\mathcal{O}(N \log N)$, with memory requirement of order $\mathcal{O}(N)$.*

Proof. Recall that P is defined as the sum

$$P = \sum_{k \in \mathcal{F}} P_k = \sum_{k \in \mathcal{F}} {\lambda_k}^{-1} A_k A_k^*. \tag{4.6.1}$$

From Proposition 4.3.1 we know that

$$A_k = \sum_{l \in \mathcal{F}_k} D_l A_N D_l^T, \tag{4.6.2}$$

where the l^{th} column of $D_l A_N D_l^T$ is the l^{th} column of A and all other columns of $D_l A_N D_l^T$ are zero. Hence

$$A_k^* = \sum_{l \in \mathcal{F}_k} D_l A_N^* D_l^T, \tag{4.6.3}$$

with the l^{th} row of $D_l A_N^* D_l^T$ being the l^{th} row of A^*, and the remaining rows being zero.

Let $a = [a(1)\ a(2)\ \dots\ a(N)]^T$ denote the generating sequence of A. The product $D_l A_N^* D_l^T v$ is the column vector with

$$[\overline{a(1)}\ \overline{a(2)} \dots \overline{a(k)}] \cdot [v(l)\ v(2l) \dots v(l\lfloor N/k \rfloor)]^T \tag{4.6.4}$$

in the l^{th} position, and zeros elsewhere (in MATLAB notation, we have $a(1 : k) * v(l : l : end).'$). From here we see that $A_k^* v$ has $[\overline{a(1)}\ \overline{a(2)} \dots \overline{a(k)}] \cdot [v(l)\ v(2l) \dots v(kl)]^T$ in the l^{th} position, for each $l \in \mathcal{F}_k$, and zeros elsewhere.

Note that the product $A_k(A_k^* v)$ is a linear combination of the non-zero columns of A_k, located in the l^{th} position for each $l \in \mathcal{F}_k$. Moreover, by definition, no two columns of A_k have non-zero terms in the same coordinate positions. From here we have that

$$\begin{bmatrix} P_k v(l) \\ P_k v(2l) \\ \vdots \\ P_k v(kl) \end{bmatrix} = \frac{1}{\lambda(k)} \begin{bmatrix} a(1) \\ a(2) \\ \vdots \\ a(k) \end{bmatrix} \begin{bmatrix} \overline{a(1)} & \overline{a(2)} & \dots & \overline{a(k)} \end{bmatrix} \begin{bmatrix} v(l) \\ v(2l) \\ \vdots \\ v(kl) \end{bmatrix} \tag{4.6.5}$$

for each $l \in \mathcal{F}_k$. To compute the product Pv we must repeat these computations for each $k \in \mathcal{F}$ and add them together.

Specifically, the product $u = Pv$ can be computed via the following algorithm.

Algorithm 4.6.3. *For input vectors a and v, compute F, F_K and λ via Algorithm 4.6.1 and initiate $u = Pv$ as a zero vector of the same size as v via $u = zeros(size(v))$;. Then compute u via*

$$
\begin{aligned}
&for\ k = 1 : length(F)\\
&\quad for\ l = F_k(k)\\
&\qquad u(l : l : end) = u(l : l : end) + (1/\lambda(k)) * a(1 : F(k)) * (a(1 : F(k))' * v(l : l : end));\\
&\quad end\\
&end
\end{aligned}
$$

At each iteration of the inner loop, the computations involve exactly k many coordinates of v and a. Computing $a(1 : F(k))' * v(l : l : end)$ requires $3k - 1$ operations, and multiplication by $1/\lambda(k)$ requires 2 additional operations (after $\lambda(k)$ has been computed with $\mathcal{O}(k)$ operations). Since $(1/\lambda(k))*(a(1 : F(k))'*v(l : l : end))$ is a scalar, multiplying it by $a(1 : F(k))$ requires k operations, and adding the result to $u(l : l : end)$ requires k more operations, amounting to $5k + 1$ operation. From here, the total number of operations is

$$\sum_{k\in\mathcal{F}} \sum_{l\in\mathcal{F}_k} (5k + 1). \tag{4.6.6}$$

Recall that $|\mathcal{F}|$ is the number of distinct bands of A, while $|\mathcal{F}_k|$ is the number of elements in each band. Thus we have that $\sum_{k\in\mathcal{F}} \sum_{l\in\mathcal{F}_k} = N$. Furthermore, since the elements of $\mathcal{F}$ are the distinct values of $\lfloor N/n \rfloor$, for $n = 1, 2, \ldots, N$, we have that

$$\sum_{k\in\mathcal{F}} \sum_{l\in\mathcal{F}_k} (5k + 1) = 5 \left(\sum_{n=1}^{N} \left\lfloor \frac{N}{n} \right\rfloor \right) + N = \mathcal{O}(N \log N). \tag{4.6.7}$$

Finally, since the algorithm computes the product $u = Pv$ directly from the generating sequence (a_n), the memory requirements are of order $\mathcal{O}(N)$. □

Analogously to Proposition 3.6.1, the foregoing proposition admits a generalization to compute the product PV, where V is a matrix of size $N \times M$.

Proposition 4.6.4. *Let $A = \sum_{n=1}^{N} a_n D_n$ be a finite D-matrix and let P be the P-matrix associated with A. Let V be matrix of size $N \times M$. Then the product PV can be evaluated via an algorithm of order $\mathcal{O}(NM \log N)$, with memory requirement of order $\mathcal{O}(NM)$.*

Proof. The statement of this Proposition becomes obvious with the subsequent algorithm:

Algorithm 4.6.5. *For input vector a and matrix V, compute F, F_K and λ via Algorithm 4.6.1 and initiate $U = PV$ as a zero matrix of the same size as V via $U = zeros(size(V))$;. Then compute U via*

$$
\begin{aligned}
&for\ k = 1 : length(F)\\
&\quad for\ l = F_k(k)\\
&\qquad U(l : l : end, :) = U(l : l : end, :) + (1/\lambda(k)) * a(1 : F(k)) * (a(1 : F(k))' * V(l : l : end, :));\\
&\quad end\\
&end
\end{aligned}
$$

The order of complexity of this algorithm, as well as its memory requirements, can be computed analogously to Proposition 3.6.3.

□

We conclude this section by noting that Algorithms 4.6.3 and 4.6.5 can be used for computations involving $\widetilde{P}$. Namely, it suffices to substitute F, F_k and λ, with their respective subsets corresponding to our choice of subset $\widetilde{\mathcal{F}}$.

Chapter 5

Software Implementation of D-matrices and P-matrices for Signal Processing Applications

In this chapter I will discuss some initial ways in which D-matrices and their associated P-matrices can be applied to signal processing. Significantly, my first step in the application of D-matrices, and their associated P-matrices, was to develop the appropriate software. In this chapter I have included the software implementation of D-transforms and P-transform, both within the framework of the general theory and within the framework of signal processing. I have also included a detailed explanation of the construction of D-transforms and P-transforms from a chosen waveform.

5.1 Software Implementation of D-Matrices

In this section I discuss the MATLAB implementation of the fast algorithms necessary for basic mathematical operations with D-matrices. When I began this work, I had at my disposal a MATLAB package on D-matrices developed by Sowa. Throughout my research I have refined some of the codes, and added new ones. In particular, I extended Sowa's software to two dimensional inputs and developed two dimensional transforms. Details of each particular contribution are recorded in the codes as well.

All fast transforms presented in this chapter use the generating sequences of D-matrices and their inverses, rather than using the full-size matrices. Given a generating sequence "gamma", we can use the function gamma2alpha to compute its Dirichlet inverse "alpha". I usually denote by "alpha" the generating sequence of a D-matrix A that I would like to use as a forward transform, and by "gamma" the generating sequence of the inverse transform A^{-1}. While at first glance it may appear counterintuitive to start with the generating sequence "gamma" of an inverse transform, when constructing a D-matrix from a waveform f, we indeed first construct the inverse transform, and then the forward transform. This certainly depends on what I choose to call forward transform. I will explain the process of generating forward and inverse D-transforms, in detail, in Section 5.3.

```
1 function alpha = gamma2alpha(gamma)
2     % Calculates the Dirichlet inverse of gamma and stores it into alpha.
3     % If input is row/column, output is row/column.
```

```
    alpha = -gamma/(gamma(1)^2);
    alpha(1) = -alpha(1);
    N = length(gamma);
    for k = 2:floor(N/2)
        alpha(2*k:k:N) = alpha(2*k:k:N) - alpha(1)*alpha(k)*gamma(2:floor(N/k
            ));
    end
end
% Copyright (c) A P Sowa, May 2011 and Feb 2012
% Copyright (c) I Vlahu, July 2016 (simplified the code)
```

The subsequent function, fast_A_Prod(alpha, u), allows us to compute the product Au directly from the generating sequence, via a fast algorithm. This function was initially written by Sowa for vector inputs. I have extended the code to allow for matrix inputs. Consequently, this function can also be used to generate the full size matrix A by applying the function to the identity matrix of size $N \times N$, namely, A = fast_A_Prod (alpha, eye(N)). Similarly, the function can be used to generate A^{-1} by replacing "alpha" with "gamma".

```
function v = fast_A_Prod(alpha, u)
    % Evaluates v = A*u FAST, where u can be a column or a matrix.
    N = length(alpha);
    if isrow(alpha)      % Ensures correct indexing.
        alpha = alpha.';
    end
    v = zeros(size(u));
    for k = 1:N
        v(k:k:N,:) = v(k:k:N,:) + alpha(1:floor(N/k))*u(k,:);
    end
end
% Copyright (c) A P Sowa, Feb 2012
% Copyright (c) I Vlahu, Sept 2016 (extended code to matrix input)
```

As the name suggests, the following function computes the product A^*u via a fast algorithm, where u can be a vector or a matrix. Similarly to above, the function fast_A_Dagger_Product(alpha,u) can be used to generate A^* (and hence A), by taking u = eye(N).

```
function v = fast_A_Dagger_Product(alpha,u)
    % Evaluates v = A'*u directly via alpha.
    N = length(alpha);
    if iscolumn(alpha)  % Ensures correct indexing.
```

```
        alpha = alpha.';
    end
    v = zeros(size(u));
    for k = 1:N
        v(k,:) = v(k,:)+conj(alpha(1:floor(N/k)))*u(k:k:end,:);
    end
end
% Copyright (c) I Vlahu, Sept 2016
```

5.2 Software Implementation of P-Matrices

Recall that P-matrices (and their inverses) are uniquely determined by the generating sequences of the D-matrices (and their inverses) that they are associated with. Hence, we again begin by choosing the generating sequences of our forward and inverse transforms. Once the generating sequences have been computed, to be able to perform any operations with P-matrices, we need to compute the orthogonal band decomposition induced by the generating sequence. The function getIndexSet(alpha) computes the index set $\mathcal{F}$, the orthogonal bands $\mathcal{F}_k$ and the diagonal of the norm matrix Λ directly from the generating sequence.

```
function [F, F_K, Lambda] = getIndexSet(alpha)
    N = length(alpha);
    K = 1:N;
    K = floor(N./K);
    % Floor decomposition.
    F = unique(K,'stable');  % 'stable' preserves the ordering (decreasing).
    l = length(F);
    F_K = cell(1,l);          % The floor classes.
    Lambda = zeros(1,l);
    for k = 1:l
        F_K{k} = find(K == F(k));
        Lambda(k) = (norm(alpha(1:F(k)))).^2;
    end
end
% Copyright (c) I Vlahu, Sept 2016
```

In the function above, the floor indices $k \in \mathcal{F}$, and the orthogonal bands $\mathcal{F}_k$, are stored in the corresponding location in F and F_K, respectively. For example, the first element of F is N, while the first element of F_K is $\{1\}$, because $\lfloor N/1 \rfloor = N$. In other words F, F_K and Lambda, enumerate the orthogonal bands according

to their location in the D-matrix, from left to right. In that sense, the first band contains the first column of the D-matrix generated by "alpha", while the last band contains the columns featuring "alpha(1)" only.

For the applications presented later in this chapter, I am interested in partial sums $\widetilde{P} = \sum_{k \in \widetilde{\mathcal{F}}} P_k$ which include all but the last "r" bands of the D-matrix. The subsequent function computes the product $\widetilde{P}u$ via a fast algorithm for a column vector or matrix u. Observe that when "r" is 0, the function computes the product Pu.

```
function v = fast_P_tilde_Product(alpha, u, r)
    % Evaluates v = P_tilde*u FAST.
    [F,F_K,Lambda] = getIndexSet(alpha);
    if isrow(alpha)
        alpha = alpha.';
    end
    v = zeros(size(u));
    l = length(F);
    if r > l                % Ensures that r does not exceed l.
        r = l;
    end
    for k = 1:l-r           % For vector input r, replace k = 1:l-r with k = r
        .
        for q = F_K{k}      % Multiplication by P_k.
            v(q:q:end,:) = v(q:q:end,:) + alpha(1:F(k))*((1/Lambda(k))*(alpha
                (1:F(k))'*u(q:q:end,:)));
        end
    end
end
% Copyright (c) I Vlahu, Sept 2016
```

Finally, fast_invP_tilde_Product(alpha, gamma, u, r) can be used to compute $\widetilde{P^{-1}}u$ via a fast algorithm, where u can be a vector or a matrix and "r" is chosen as above. Note that, by virtue of Theorem 4.5.7, the output of fast_invP_tilde_Product(alpha, gamma, fast_P_tilde_Product(alpha, u, r), s); is the same for any "s" less than or equal to "r". Choosing "s" to be equal to "r," is therefore, computationally most efficient.

```
function v = fast_invP_tilde_Product(alpha, gamma, u, r)
    % Evaluates v = invP*u FAST.
    [~,F_K,Lambda] = getIndexSet(alpha);
    u = fast_A_Prod(gamma, u);
    l = length(F_K);
```

```
    if r > l                % Ensures that r does not exceed l.
        r = l;
    end
    for k = 1:l-r          % For vector input r, replace k = 1:l-r with k = r.
        u(F_K{k},:) = Lambda(k).*u(F_K{k},:);
    end
    v = fast_A_Dagger_Product(gamma,u);
end
% Copyright (c) I Vlahu, Sept 2016 and Oct 2017
```

As in the case of D-matrices, the preceding two functions can be used to generate matrices $\widetilde{P}$ and $\widetilde{P^{-1}}$ by evaluating the corresponding functions for input u = eye(N).

5.3 D-transforms for Signal Processing

In Section 2.3, I briefly discussed Sowa's construction of a D-transform from a waveform f. In this section I revisit this construction to discuss possible modifications together with my extension of the one dimensional D-transform to a two dimensional D-transform. I will also discuss the software implementation of the D-transforms within the context of signal processing.

Recall that a D-matrix (or a P-matrix) of size $N \times N$ is defined by a sequence of N numbers. Throughout the remainder of this chapter, I will denote by (α_n) the generating sequence of the forward transforms, and by (γ_n) the generating sequence of the inverse transforms. By forward transforms, I will refer to change of basis transforms from the Fourier basis representation to a D-basis (or a P-basis) representation. Similarly, by an inverse transform I will refer to change of basis transforms from a D-basis (or a P-basis) representation to the Fourier basis representation.

Sowa's construction of a D-basis, as described in Section 2.3, uses a waveform (function) $w \in L^2[0, 2\pi]^{2N+1}$ to generate, what I call, an *inverse* D-transform of size $N \times N$. The construction uses the positive-frequency coefficients of w to define the function $f(t) = \sum_{n=1}^{N} \gamma_n e^{int} \in H_h^N$, where $(\gamma_n)_{n=-N}^{N}$ denote the Fourier

coefficients of w obtained via $\frac{1}{2N+1}\text{fft}(w)$. The D-matrix

$$A^{-1} = \begin{bmatrix} \gamma_1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \gamma_2 & \gamma_1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \gamma_3 & \cdot & \gamma_1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \gamma_4 & \gamma_2 & \cdot & \gamma_1 & \cdot & \cdot & \cdot & \cdot \\ \gamma_5 & \cdot & \cdot & \cdot & \gamma_1 & \cdot & \cdot & \cdot \\ \gamma_6 & \gamma_3 & \gamma_2 & \cdot & \cdot & \gamma_1 & \cdot & \cdot \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \cdot \\ \gamma_N & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \gamma_1 \end{bmatrix}, \tag{5.3.1}$$

is the change of basis matrix from the representation in the D-basis $\{f(nt)\}_{n=1}^N$ to the Fourier representation (within the positive-frequency subspace H_h^N). Given the Fourier coefficients of a signal, to obtain the D-basis representation, we need to apply the inverse matrix

$$A = \begin{bmatrix} \alpha_1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \alpha_2 & \alpha_1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \alpha_3 & \cdot & \alpha_1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \alpha_4 & \alpha_2 & \cdot & \alpha_1 & \cdot & \cdot & \cdot & \cdot \\ \alpha_5 & \cdot & \cdot & \cdot & \alpha_1 & \cdot & \cdot & \cdot \\ \alpha_6 & \alpha_3 & \alpha_2 & \cdot & \cdot & \alpha_1 & \cdot & \cdot \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \cdot \\ \alpha_N & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \alpha_1 \end{bmatrix}, \tag{5.3.2}$$

where (α_n) is the Dirichlet inverse of (γ_n). The process of generating an inverse D-transform (i.e., a generating sequence (γ_n)) from a waveform w is summarized in the following MATLAB code:

```
function gamma = wave2gamma(wave)
    % Extracts the positive-frequency coefficients of the waveform 'wave'.
    % The digital length of 'wave' (vector dimension) should be the same as
        that of the signals to be processed.
    n = length(wave);
    scale = 1/n;
    coefs = scale*fft(wave);
    halflength = floor(n/2);
    gamma = coefs(2:halflength+1);
end
% Copyright (c) A P Sowa, May 2011
```

To generate the corresponding forward transform, we use the function gamma2alpha(gamma) introduced in Section 5.1.

Once the generating sequences of the forward and inverse transforms have been computed, the next step is to apply them to signals. Since forward D-transforms (or P-transforms) represent change of basis transforms from the Fourier basis representation to a D-basis (or P-basis) representation, within the positive-frequency subspace, the first step is to isolate the positive-frequency coefficients of the signal we are working with. Given a signal $\mathfrak{I}$ in the time domain (for one dimensional signals) or the spatial domain (for images), with column length $2N+1$, the first step is to apply the Fast Fourier Transform to obtain the Fourier coefficients. We then apply D-transforms (or P-transforms) of size $N \times N$ to the positive-frequency coefficients and complex conjugates of D-transforms (or P-transforms) to the negative-frequency coefficients independently. The zero-frequency remains unaffected. We may choose to use the same transform for the positive-frequency coefficients and the negative-frequency coefficients. This corresponds to an extension of the basis $\{f(nt)\}_{n=1}^{N}$ for the positive-frequency subspace to the basis $\{1\} \cup \{f(nt), \overline{f(nt)}\}_{n=1}^{N}$ of the entire $L^2[0, 2\pi]^{2N+1}$ space. Alternatively, we could choose a transform generated by some f for the positive-frequency coefficients and another transform generated by some g for the negative-frequency coefficients, corresponding to the basis $\{1\} \cup \{f(nt), \overline{g(nt)}\}_{n=1}^{N}$. Depending on the particular application, one approach could be more appropriate than the other.

Before we proceed to the codes, it is important to note that the output of the MATLAB built-in Fast Fourier Transform orders the Fourier coefficients by placing the zero-frequency coefficient first, then the positive-frequency coefficients, and finally the negative-frequency coefficients. For a signal of (column) length $2N+1$, the output has an equal number of positive-frequency and negative-frequency coefficients. If the signal is real, the k^{th} and $((2N+1)-(k-2))^{\text{th}}$ column entries are conjugates of each other for $k = 2, \ldots, N+1$. Alternatively, for a signal of length $2N$, the output includes N positive-frequency coefficients and $N-1$ negative-frequency coefficients. In the latter care, if the signal is real, the conjugate symmetry extends to the first $N-1$ positive-frequency coefficients, i.e., the k^{th} and $(2N-(k-2))^{\text{th}}$ column entries are conjugates of each other for $k = 2, \ldots, N$.

If we wish to view the Fourier coefficients in the standard ordering (first negative–frequency, then zero, then positive-frequency) we can apply the MATLAB function shiftfft(fft(signal)). The conjugate symmetry will then be about the zero-frequency coefficient located at the $(N+1)^{\text{st}}$ position (in the case of signals of length $2N+1$), or the N^{th} position (in the case of signals of length $2N$).

5.3.1 D-transforms Generated by One Waveform

When applying D-transforms generated by one waveform, to the Fourier coefficients of a signal (with the standard ordering), we are essentially multiplying by a matrix of the form:

$$\begin{bmatrix}
\ddots & \vdots & \vdots & \vdots & \vdots & & & & & \\
 & \overline{\alpha}_1 & & \overline{\alpha}_2 & \overline{\alpha}_4 & & & & & \\
 & & \overline{\alpha}_1 & & \overline{\alpha}_3 & & & & & \\
 & & & \overline{\alpha}_1 & \overline{\alpha}_2 & & & & & \\
 & & & & \overline{\alpha}_1 & & & & & \\
 & & & & & 1 & & & & \\
 & & & & & & \alpha_1 & & & \\
 & & & & & & \alpha_2 & \alpha_1 & & \\
 & & & & & & \alpha_3 & & \alpha_1 & \\
 & & & & & & \alpha_4 & \alpha_2 & & \alpha_1 \\
 & & & & & & \vdots & \vdots & \vdots & \vdots
\end{bmatrix}. \tag{5.3.3}$$

Clearly we do not want to store this matrix, but rather work with the generating sequence (α_n) instead. And indeed, we apply the D-matrix A (or rather the sequence (α_n)) to the positive-frequency coefficients of the signal, while we apply $\overline{A}$ (i.e., the sequence $(\overline{\alpha}_n)$) to the negative-frequency coefficients (after appropriate reordering).

In the case of real signals, we can take advantage of the fact that the Fourier coefficients exhibit conjugate symmetry. Namely, if $\phi(t) = \sum_n c_n e^{int}$ is a real valued function, then we have that

$$\sum_n c_n e^{int} = \phi(t) = \overline{\phi(t)} = \sum_n \overline{c_n} e^{-int} = \sum_n \overline{c_{-n}} e^{int}. \tag{5.3.4}$$

To transform a real signal using a basis of the form $\{1\} \cup \{f(nt), \overline{f(nt)}\}_{n=1}^{N}$, it is not necessary that both sets of coefficients are transformed. Instead, it suffices to extract and transform the positive-frequency component of the signal, and then use conjugate symmetry to retrieve the transformed negative-frequency coefficients. To transform a complex input, on the other hand, both the positive-frequency and the negative-frequency components must be extracted and independently transformed.

One Dimensional D-transforms

The fastDirichletTransform(signal, alpha) below was initially written by Sowa for real vector inputs. I later extended the function to complex matrix inputs. When the input is a matrix, the transform is applied to each column, as expected. Note that this is still a one-dimensional transform as it operates on one dimension, namely on the columns of the input signal. Finally, the function is based on a basis of the form $\{1\} \cup \{f(nt), \overline{f(nt)}\}_{n=1}^{N}$ or $\{1\} \cup \{f(nt)\}_{n=1}^{N} \cup \{\overline{f(nt)}\}_{n=1}^{N-1}$, depending on the column length of the signal.

```
function Transform_Coefs = fast_Dirichlet_Transform(signal,alpha)
    % Computes the transform coefficients directly via alpha.
    % The transform operates on the columns of a signal.

    n = size(signal,1);                % n is the length of each column.
    halfLength = floor(n/2);

    scale = 1/n;
    coefs = scale*fft(signal);         % The Fourier coefficients of the signal.

    holo = coefs(2:halfLength+1,:);  % Extract positive-frequency component.
    trans_holo = Dirichlet_Fast_Prod(alpha, holo); % Apply fast transform.

    % If the signal is real, take advantage of the conjugate symmetry between
       the positive-frequency and negative-frequency coefficients.
    if isreal(signal)
        % Restore the entire set of coefficients.
        if halfLength == n/2
            Transform_Coefs = [coefs(1,:); trans_holo; flipud(conj(trans_holo
               (1:end-1,:)))];
        else
            Transform_Coefs = [coefs(1,:); trans_holo; flipud(conj(trans_holo)
               )];
        end
    else
        anti_holo = coefs(halfLength+2:end,:); % Extract anti-holo component.
        % Define fast transform for anti-holo component of correct length.
        if halfLength == n/2
            alpha_bar = conj(alpha(1:end-1));
        else
            alpha_bar = conj(alpha);
        end
        % Apply fast transform.
        trans_anti_holo = Dirichlet_Fast_Prod(alpha_bar, flipud(anti_holo));
        % Restore the entire set of coefficients.
```

```
        Transform_Coefs = [coefs(1,:); trans_holo; flipud(trans_anti_holo)];
    end
end
% Copyright (c) A P Sowa, Feb 2012 (real vector inputs)
% Copyright (c) I Vlahu, March 2014 (complex matrix inputs)
```

Next, I discuss inverse D-transforms. Again, we can take advantage of conjugate symmetry. Observe that if we have a read signal, the conjugate symmetry exhibited by the Fourier coefficients is also present in the D-basis coefficients (by construction). Hence, to take advantage of conjugate symmetry when applying the inverse transform, we must either make assumptions on the input, or apply a test on the coefficients. Sowa's inverse D-transform was originally constructed for real one dimensional signals using conjugate symmetry. I generalized Sowa's function to one that tests for conjugate symmetry and applies the inverse transforms accordingly.

```
function Signal_Reconstruct = inv_fast_Dirichlet_Transform(Transform_Coefs,
    gamma)
    % Computes the inverse transform FAST directly via gamma.
    % There are no assumptions of conjugate symmetry.

    n = size(Transform_Coefs,1); % n is the length of each column.
    halflength = floor(n/2);

    trans_holo = Transform_Coefs(2:halflength+1,:);
    % Restore positive-frequency Fourier coefficients.
    holo = Dirichlet_Fast_Prod(gamma, trans_holo);

    if halflength == n/2
        % Test for conjugate symmetry.
        if trans_holo(1:halflength-1,:) == flipud(conj(Transform_Coefs(
            halflength+2:end,:)));
            % Use conjugate symmetry to restore all coefficients.
            coefs = [Transform_Coefs(1,:); holo; flipud(conj(holo(1:halflength
                -1,:)))];
        else
            % Restore negative-frequency coefficients.
            trans_anti_holo = Transform_Coefs(halflength+2:end,:);
            % Choose transform of correct length.
```

```
            gamma_bar = conj(gamma(1:end-1));
            % Apply transform.
            anti_holo = Dirichlet_Fast_Prod(gamma_bar, flipud(trans_anti_holo)
                );
            % Restore all coefficients.
            coefs = [Transform_Coefs(1,:); holo; flipud(anti_holo)];
        end
    else
        % Test for conjugate symmetry.
        if trans_holo == flipud(conj(Transform_Coefs(halflength+2:end,:)));
           % Use conjugate symmetry to restore all coefficients.
            coefs = [Transform_Coefs(1,:); holo; flipud(conj(holo(1:halflength
                ,:)))];
        else
            % Restore negative-frequency Fourier coefficients.
            trans_anti_holo = Transform_Coefs(halflength+2:end,:);
            % Choose transform.
            gamma_bar = conj(gamma);
            % Apply transform.
            anti_holo = Dirichlet_Fast_Prod(gamma_bar, flipud(trans_anti_holo)
                );
            % Restore all coefficients.
            coefs = [Transform_Coefs(1,:); holo; flipud(anti_holo)];
        end
    end
    % Perform ifft to restore time/spatial domain coefficients.
    coefs = n*coefs;
    Signal_Reconstruct = ifft(coefs);
end
% Copyright (c) A P Sowa, May 2011 and Feb 2012
% Copyright (c) I Vlahu, Jan 2014 and Oct 2017
```

Two Dimensional D-Transforms

Having defined the one dimensional transforms, the two dimensional transform is achieved by repeatedly applying the one dimensional transform first to the columns and then to the rows of the signal. This

construction is analogous to the two dimensional Fourier Transform.

```
function Transform_Coefs = fast_Dirichlet_Transform_2D(signal, alpha)
    % Computes the two dimensional forward D-transform FAST.
    Transform_Coefs = fast_Dirichlet_Transform(fast_Dirichlet_Transform(signal
        ,alpha).',alpha).';
end
% Copyright (c) I Vlahu, Jan 2014
```

Note that the two dimensional inverse transform must first transform the rows and then the columns. Note also, that the inverse transform assumes real outputs.

```
function Signal_Reconstruct_2D = inv_fast_Dirichlet_Transform_2D(
    Transform_Coefs_2D, gamma)
    % Computes the two dimensional inverse D-transform FAST.
    Signal_Reconstruct_2D = inv_fast_Dirichlet_Transform(
        inv_fast_Dirichlet_Transform(Transform_Coefs_2D.',gamma).',gamma);
    % Remove possible complex residue.
    Signal_Reconstruct_2D = real(Signal_Reconstruct_2D);
end
% Copyright (c) I Vlahu, Jan 2014
```

5.3.2 D-transforms Generated by Two Waveforms

One Dimensional D-transforms

As noted in Section 5.3, we can choose to transform the positive-frequency and negative-frequency coefficients of a signal with two different D-transforms (and P-transforms). Such transforms would correspond to a basis $\{1\} \cup \{f(nt), \overline{g(nt)}\}$ generated by two waveforms. If (α_n) denotes the generating sequence of the forward D-transform induced by f and (β_n) denotes the generating sequence of the forward D-transform induced by g, then applying the corresponding D-transform, to the Fourier coefficients of a signal (with the standard

ordering), represents multiplication by a matrix of the form:

$$\begin{bmatrix}
\ddots & \vdots & \vdots & \vdots & \vdots & & & & & \\
 & \overline{\beta}_1 & & \overline{\beta}_2 & \overline{\beta}_4 & & & & & \\
 & & \overline{\beta}_1 & & \overline{\beta}_3 & & & & & \\
 & & & \overline{\beta}_1 & \overline{\beta}_2 & & & & & \\
 & & & & \overline{\beta}_1 & & & & & \\
 & & & & & 1 & & & & \\
 & & & & & & \alpha_1 & & & \\
 & & & & & & \alpha_2 & \alpha_1 & & \\
 & & & & & & \alpha_3 & & \alpha_1 & \\
 & & & & & & \alpha_4 & \alpha_2 & & \alpha_1 \\
 & & & & & & \vdots & \vdots & \vdots & \vdots & \ddots
\end{bmatrix}. \tag{5.3.5}$$

Again, we do not store this matrix, but rather use the generating sequence (α_n) to transform the positive-frequency coefficients of the signal, while the negative-frequency coefficients are transformed using $(\overline{\beta}_n)$. Note that in this case the conjugate symmetry of real signals is not used.

```
function Transform_Coefs = fast_Dirichlet_Transform_2_waves(signal, alpha,
    beta)
    % Computes the transform coefficients directly via alpha and beta.
    % The transform operates on the columns of a signal.

    n = size(signal,1);                % n is the length of each column.
    halfLength = floor(n/2);

    scale = 1/n;
    coefs = scale*fft(signal);         % The Fourier coefficients of the signal.

    holo = coefs(2:halfLength+1,:);  % Extract positive-frequency component.
    trans_holo = Dirichlet_Fast_Prod(alpha, holo); % Apply fast transform.

    anti_holo = coefs(halfLength+2:end,:); % Extract anti-holo component.
    if halfLength == n/2
        beta_bar = conj(beta(1:end-1));
    else
        beta_bar = conj(beta);
    end
    trans_anti_holo = Dirichlet_Fast_Prod(beta_bar, flipud(anti_holo));
```

```
    % Apply fast transform.
    trans_anti_holo = flipud(trans_anti_holo);

    % Restore the entire set of coefficients.
    Transform_Coefs = [coefs(1,:); trans_holo; trans_anti_holo];
end
% Copyright (c) I Vlahu, Oct 2017
```

To apply the inverse transform, we need to have the corresponding generating sequences. In the function below, "delta" denotes the Dirichlet inverse of "beta".

```
function Signal_Reconstruct = inv_fast_Dirichlet_Transform_2_waves(
    Transform_Coefs, gamma, delta)
    % Computes the inverse transform directly via gamma.
    % Makes no assumptions on output symmetry.

    n = size(Transform_Coefs,1); % n is the length of each column.
    halflength = floor(n/2);

    % Extract positive-frequency AND negative-frequency components.
    trans_holo = Transform_Coefs(2:halflength+1,:);
    trans_anti_holo = Transform_Coefs(halflength+2:end,:);

    % Restore positive-frequency Fourier coefficients.
    holo = Dirichlet_Fast_Prod(gamma, trans_holo);

    if halflength == n/2
        delta_bar = conj(delta(1:end-1));
    else
        delta_bar = conj(delta);
    end
    % Restore negative-frequency Fourier coefficients.
    anti_holo = Dirichlet_Fast_Prod(delta_bar, flipud(trans_anti_holo));
    anti_holo = flipud(anti_holo);

    % Obtain Fourier coefficients.
    coefs = [Transform_Coefs(1,:); holo; anti_holo];
```

```
    % Perform ifft to restore time/spatial domain coefficients.
    coefs = n*coefs;
    Signal_Reconstruct = ifft(coefs);
end
% Copyright (c) I Vlahu, Oct 2017
```

Two Dimensional D-transforms

The two dimensional transforms generated by one waveform can easily be adapted to transforms generated by two waveforms. We simply need to add the generating sequences "beta" and "gamma" to the list of inputs.

```
function Transform_Coefs = fast_Dirichlet_Transform_2D_2_waves(signal, alpha,
    beta)
    Transform_Coefs = fast_Dirichlet_Transform_2_waves(
        fast_Dirichlet_Transform_2_waves(signal,alpha, beta).',alpha, beta).';
end
% Copyright (c) I Vlahu, October 2017
```

```
function Signal_Reconstruct_2D = inv_fast_Dirichlet_Transform_2D_2_waves(
    Transform_Coefs_2D, gamma, delta)
    Signal_Reconstruct_2D = inv_fast_Dirichlet_Transform_2_waves(
        inv_fast_Dirichlet_Transform_2_waves(Transform_Coefs_2D.',gamma, delta)
        .',gamma, delta);
    % Remove possible complex residue.
    Signal_Reconstruct_2D = real(Signal_Reconstruct_2D);
end
% Copyright (c) I Vlahu, Oct 2017
```

5.4 P-transforms for Signal Processing

Recall that, given a D-matrix A, the associated P-matrix P is defined as the sum of orthogonal projections onto the orthogonal bands of A. Namely, the matrix P is a sum of projections onto subspaces of the column space of A, i .e., the D-basis. Hence, P-transforms are also to be applied to the Fourier coefficients of signals. In practical terms, P-transforms can be applied in place of D-transforms. Furthermore, the generating sequences for D-matrices also generate P-matrices. As already noted, the resulting transforms are again fast.

5.4.1 P-transforms Generated by One Waveform

As in the case of D-transforms, P-transforms are independently applied to the positive-frequency and the negative-frequency coefficients of signals. Hence, the potential to transform the positive-frequency and negative-frequency coefficients using the same waveform, or two different waveforms, remains. First we consider P-transforms generated by a single waveform.

The One Dimensional P-transforms

The one dimensional forward P-transform is defined analogously to the one dimensional forward D-transform. Namely, the signal is first transformed using the Fast Fourier Transform to obtain the Fourier coefficients and then the P-transform is applied. For real signals, the transform again takes advantage of conjugate symmetry, which in turn results in conjugate symmetry in the output as well. Significantly, the function fast_P_Tilde_Transform(alpha, signal, r) can be used for any matrix $\widetilde{P}$ by specifying, in "r," the number of orthogonal projections to be excluded. One could also pass a vector of indices for the input variable "r" provided that the function fast_P_tilde_Product(alpha, u, r) has been adapted to such a variable. Note that if any orthogonal projection is excluded, then the transform is non-invertible. We will see in Section 5.6 how the following two functions can be used to filter signals.

```
function Transform_Coefs = fast_P_tilde_Transform(alpha, signal, r)
    % Computes the transform coefficients FAST directly via alpha.
    % r defines P_Tilde.

    n = size(signal,1); % n is the length of each column.
    halfLength = floor(n/2);

    scale = 1/n;
    coefs = scale*fft(signal); % The Fourier coefficients of the signal.

    holo = coefs(2:halfLength+1,:); % Extract positive-frequency component.
    trans_holo = fast_P_tilde_Product(alpha,holo,r); % Apply fast transform.

    if isreal(signal)
        % Restore the entire set of coefficients using conjugate symmetry.
        if halfLength == n/2
            Transform_Coefs = [coefs(1,:); trans_holo; flipud(conj(trans_holo
                (1:end-1,:)))];
        else
```

```
            Transform_Coefs = [coefs(1,:); trans_holo; flipud(conj(trans_holo)
                )];
        end
    else
        % Transform negative-frequency component.
        anti_holo = coefs(halfLength+2:end,:);
        % Choose transform of correct length.
        if halfLength == n/2
            alpha_bar = conj(alpha(1:end-1));
        else
            alpha_bar = conj(alpha);
        end
        % Apply transform.
        trans_anti_holo = fast_P_tilde_Product(alpha_bar,flipud(anti_holo),r);
        % Restore the entire set of coefficients.
        Transform_Coefs = [coefs(1,:); trans_holo; flipud(trans_anti_holo)];
    end
end
% Copyright (c) I. Vlahu, Sept 2016
```

As hinted above, the inverse P-transform reverses the action of the forward P-transform as long as no orthogonal projections have been excluded.

```
function Signal_Reconstruct = inv_fast_P_tilde_Transform(alpha, gamma,
    Transform_Coefs,r)
    % Computes the inverse transform FAST directly via alpha and gamma.

    n = size(Transform_Coefs,1); % n is the length of each column.
    halflength = floor(n/2);

    trans_holo = Transform_Coefs(2:halflength+1,:);
    holo = fast_invP_tilde_Product(alpha, gamma, trans_holo,r); % Apply
        transform.

    if halflength == n/2
        % Test for conjugate symmetry.
```

```
12          if trans_holo(1:halflength-1,:) == flipud(conj(Transform_Coefs(
                halflength+2:end,:)));
13              % Use conjugate symmetry to restore all coefficients.
14              coefs = [Transform_Coefs(1,:); holo; flipud(conj(holo(1:halflength
                    -1,:)))];
15          else
16              % Transform negative-frequency coefficients.
17              trans_anti_holo = Transform_Coefs(halflength+2:end,:);
18              % Choose transforms of correct length.
19              alpha_bar = conj(alpha(1:end-1));
20              gamma_bar = conj(gamma(1:end-1));
21              % Apply transform.
22              anti_holo = fast_invP_tilde_Product(alpha_bar, gamma_bar, flipud(
                    trans_anti_holo),r);
23              % Restore all coefficients.
24              coefs = [Transform_Coefs(1,:); holo; flipud(anti_holo)];
25          end
26      else
27          % Test for conjugate symmetry.
28          if trans_holo == flipud(conj(Transform_Coefs(halflength+2:end,:)));
29              % Use conjugate symmetry to restore all coefficients.
30              coefs = [Transform_Coefs(1,:); holo; flipud(conj(holo(1:halflength
                    ,:)))];
31          else
32              % Transform negative-frequency coefficients.
33              trans_anti_holo = Transform_Coefs(halflength+2:end,:);
34              % Choose transforms of correct length.
35              alpha_bar = conj(alpha);
36              gamma_bar = conj(gamma);
37              % Apply transform.
38              anti_holo = fast_invP_tilde_Product(alpha_bar, gamma_bar, flipud(
                    trans_anti_holo),r);
39              % Restore all coefficients.
40              coefs = [Transform_Coefs(1,:); holo; flipud(anti_holo)];
41          end
42      end
```

```
    % Perform ifft to restore time/spatial domain coefficients.
    coefs = n*coefs;
    Signal_Reconstruct = ifft(coefs);
end
% I Vlahu, Sept 2016 and Oct 2017
```

The Two Dimensional P-transforms

Again, the two dimensional forward and inverse P-transforms are defined analogously to the corresponding D-transforms.

```
function transformedSignal2D = fast_P_tilde_Transform_2D(alpha, signal, r)
    transformedSignal2D = fast_P_tilde_Transform(alpha, fast_P_tilde_Transform
        (alpha, signal, r).',r).';
end
% Copyright (c) I Vlahu, Sept 2016
```

```
function reconstructedSignal2D = inv_fast_P_tilde_Transform_2D(alpha, gamma,
    transformedSignal2D, r)
    reconstructedSignal2D = inv_fast_P_tilde_Transform(alpha, gamma,
        inv_fast_P_tilde_Transform_general(alpha, gamma, transformedSignal2D.',
        r ).',r);
    reconstructedSignal2D = real(reconstructedSignal2D);
end
% Copyright (c) I Vlahu, Sept 2016
```

5.4.2 P-transforms Generated by Two Waveforms

Similarly to the case of D-transform, the P-transforms generated by two waveforms are obtained with minor modifications to their one-waveform counterparts.

One Dimensional P-transforms

Note that in addition to the second generating sequence "beta", the list of inputs includes an "m" as well. This "m" defines the $\widetilde{P}$ matrix corresponding to the generating sequence "beta". Namely, if we wish to use the subsequent function as a filter (to change our signal), we can choose to transform the positive-frequency and the negative-frequency coefficients with a different set of band projections.

```
function Transform_Coefs = fast_P_tilde_Transform_2_waves(alpha, beta, signal,
    r, m)
    % Computes the transform coefficients directly via alpha.
    % The transform operates on the columns of a signal.
    % r is the number of bands to be discarded from the alpha generated P-
        matrix.
    % m is the number of bands to be discarded from the beta generated P-
        matrix.

    % get the Fourier coefficients of the signal
    n = size(signal,1); % n is the length of each columns
    halfLength = floor(n/2);

    scale = 1/n;
    coefs = scale*fft(signal);

    holo = coefs(2:halfLength+1,:);
    anti_holo = coefs(halfLength+2:end,:);
    anti_holo = flipud(anti_holo);

    % Transform positive-frequency coefficients using alpha.
    trans_holo = fast_P_tilde_Product(alpha,holo,r);

    % Transform negative-frequency coefficients using beta.
    if halfLength == n/2
        beta_bar = conj(beta(1:end-1));
    else
        beta_bar = conj(beta);
    end
    trans_anti_holo = fast_P_tilde_Product(beta_bar,anti_holo,m);
    trans_anti_holo = flipud(trans_anti_holo);
    % Restore the entire set of coefficients.
    Transform_Coefs = [coefs(1,:); trans_holo; trans_anti_holo];
end
% Copyright (c) I. Vlahu, Sept 2016 and Oct 2017
```

As in the case of P-transforms generated by one waveform, if the forward transform excludes certain orthogonal bands, then the same can be done in the inverse P-transform as well to improve computational efficiency.

```
function Signal_Reconstruct = inv_fast_P_tilde_Transform_2_waves(alpha, beta,
    gamma, delta, Transform_Coefs, r, m)
    % Computes the inverse transform directly via gamma

    n = size(Transform_Coefs,1); % n is the length of each column.
    halflength = floor(n/2);
    trans_holo = Transform_Coefs(2:halflength+1,:);
    % Apply fast transform.
    holo = fast_invP_tilde_Product(alpha, gamma, trans_holo,r);
    trans_anti_holo = Transform_Coefs(halflength+2:end,:);
    trans_anti_holo = flipud(trans_anti_holo);
    % Apply fast transform.
    if halflength == n/2
        delta_bar = conj(delta(1:end-1));
        beta_bar = conj(beta(1:end-1));
    else
        delta_bar = conj(delta);
        beta_bar = conj(beta);
    end
    anti_holo = fast_invP_tilde_Product(beta_bar, delta_bar, trans_anti_holo,m
        );
    anti_holo = flipud(anti_holo);
    % Obtain Fourier coefficients.
    coefs = [Transform_Coefs(1,:); holo; anti_holo];
    % Perform ifft to restore time/spatial domain coefficients.
    coefs = n*coefs;
    Signal_Reconstruct = ifft(coefs);
end
% Copyright (c) I. Vlahu, Sept 2016 and Oct 2017
```

Two Dimensional P-transforms

Again, the generalization to two waveforms is achieved with the addition of the new generating sequences and the corresponding band choices.

```
function transformedSignal2D = fast_P_tilde_Transform_2D_2_waves(alpha, beta,
    signal, r, m)
    transformedSignal2D = fast_P_tilde_Transform_2_waves(alpha, beta,
        fast_P_tilde_Transform_1_waves(alpha, beta, signal, r, m).', r, m).';
end
% Copyright (c) I Vlahu, Sept 2016 and Oct 2017
```

```
function reconstructedSignal2D = inv_fast_P_tilde_Transform_2D_2_waves(alpha,
    beta, gamma, delta, transformedSignal2D, r, m)
    reconstructedSignal2D = inv_fast_P_tilde_Transform_2_waves(alpha, gamma,
        inv_fast_P_tilde_Transform_2_waves(alpha, beta, gamma, delta,
        transformedSignal2D.', r, m).', r, m);
    reconstructedSignal2D = real(reconstructedSignal2D);
end
% Copyright (c) I Vlahu, Sept 2016 and Oct 2017
```

5.5 D-transforms and P-transforms Generated by Waveforms of Half-length

We discussed, in Section 2.3, Sowa's work on how D-matrices (under certain conditions) arise as change of basis transforms that mitigate between representations of functions in the Fourier basis and representations in bases of the form $\{f(nt)\}_{n=1}^{\infty}$, within the positive-frequency subspace of $L^2[0, 2\pi]$. The compatibility of D-matrices, with truncation to upper left corners, allows us to apply the theory of the infinite D-matrices in the finite discrete case. When discussing the finite framework, however, the conditions on the choice of D-matrices reduce to a simple condition on invertibility. In other words, any invertible D-matrix is a change of basis matrix, and the same is true for the associated P-matrices.

We noted in Section 5.3, that when we apply D-matrices (or P-matrices) to signals of column length $2N + 1$ (or $2N$ with minor adjustments), we use D-transforms of size $N \times N$ to independently transform (unless we choose to use conjugate symmetry) the positive-frequency and negative-frequency components of signals. Recall that we generated these D-transforms from the positive-frequency coefficients of a waveform $w(t) \in L^2[0, 2\pi]^{2N+1}$ (or $w(t) \in L^2[0, 2\pi]^{2N}$). We could, however, generate D-transforms of size $N \times N$ by using a waveform $\omega(t) \in L^2[0, 2\pi]^N$ evaluated at half (or rather floor of the half) as many points in the

interval $[0, 2\pi]$. In such a case, the generating sequence of the inverse transform would consist of all Fourier coefficients of ω, rather than just the positive-frequency ones. All codes for D-transforms and P-transforms, whether based on one or two waveform bases, would work in the same manner. The only necessary adaptation would be in the function wave2gamma. Namely, one would need to use the following function to precompute "gamma":

```
function gamma = half_wave2gamma(wave)
   % The digital length of 'wave' (vector dimension) should be HALF of the (
       column) length of the signals to be processed.
   n = length(wave);
   scale = 1/n;
   gamma = scale*fft(wave);
end
% Copyright (c) I Vlahu, Sep 2016
```

5.6 P-matrices as Filters

Let A be D-matrix of size $N \times N$ and let P be its associated P-matrix. Consider a subset $\widetilde{\mathcal{F}} \subset \mathcal{F}$ of orthogonal bands corresponding to the floor decomposition of N. Let $\widetilde{A}$, $\widetilde{I}$ and $\widetilde{P}$ and $\widetilde{P^{-1}}$ be as in Section 4.5. Let v denote the positive-frequency coefficients of a signal.

Recall that a filter acts on a signal to produce a modified signal. As noted in Section 2.2.2, in the frequency domain, filtering amounts to multiplication. For example, if we have a signal represented in the Fourier basis, we can preserve certain frequencies and discard others via multiplication by a diagonal matrix $\overline{I}$, with ones along the diagonal in the columns corresponding to the frequencies that we wish to preserve, and zeros elsewhere. Namely, $\overline{v} = \overline{I}v$ would be the filtered signal.

Recall that the Fourier basis is the D-basis corresponding to the identity matrix. To generalize the filter above to an arbitrary D-matrix A, we can preserve certain columns of A while setting the remaining ones to 0. This amounts to multiplication by $\overline{A} = A\overline{I}$. To then restore the Fourier representation, we need to multiply by A^{-1}. In other words, we would have

$$\overline{v}_A = A^{-1}\overline{A}v = A^{-1}A\overline{I}v = \overline{I}v = \overline{v}. \tag{5.6.1}$$

Hence we see that, filtering in the D-basis via the filter $A^{-1}\overline{A}$ represents filtering in the Fourier basis via $\overline{I}$.

Curiously, if we replace $\overline{A}$ with $\widetilde{P}$ (and A^{-1} with P^{-1}), we obtain the filter:

$$F = P^{-1}\widetilde{P} = A\widetilde{I}A^{-1} = \widetilde{A}A^{-1}. \tag{5.6.2}$$

Observe that F is idempotent by virtue of Corollary 4.5.8. To better understand how this filter affects signals, we can turn to numerical experimentation.

5.6.1 1D Examples

In the examples to follow, we will consider 1D signals of digital length equal to 512 and we will apply to them filters of the form $P^{-1}\widetilde{P}$. To transform signals of length 512, we require P-transforms of size 256×256 to transform the positive-frequency coefficients of the signals, and we require P-transforms of size 255×255 to transform the negative-frequency coefficients of the signals. As noted in Section 5.5, we can generate these transforms using the positive-frequency coefficients of a waveform sampled at 512 points via wave2gamma(wave), or we can use all Fourier coefficients of a waveform sampled at 256 points to generate the transforms via half_wave2gamma(wave). To indicate a P-transform generated with positive-frequency coefficients via wave2gamma(wave), I will denote the generating waveform by w_{512}. Analogously, to indicate a P-transform generated via half_wave2gamma(wave) with all Fourier coefficients, I will denote the generating waveform by w_{256}. In all subsequenct examples, "r" will denote the number of bands excluded from the sum of projections as described in Section 5.2, thus defining $\widetilde{P}$.

The Dirac delta function

Mathematically, an impulse is represented by the Dirac delta function. Hence to study the filter profile of $P^{-1}\widetilde{P}$, for a given generating waveform, the idea is to apply it to the Dirac delta function.

In Figures 5.1 and 5.2 we will consider the effects of the filters $F = P^{-1}\widetilde{P}$, for r = 1, 2, 3, 4 and 5, generated using the same waveform $w = \sin(4t)e^{-(t-\pi)^2} + 1$ for $t \in [0, 2\pi]$, on the Dirac delta function. In Figure 5.1 the graphs are generated using w_{512}, while in Figure 5.2 the graphs are generated using w_{256}. In the first case, the removal of even the last (highest frequency) orthogonal band introduces significant distortion to the Dirac delta function. Alternatively, when using w_{256} to generate the P-transforms, the filter preserves the impulse characterization of the Dirac delta function even with the removal of several bands.

Interestingly enough, repeating the experiments of Figures 5.1 and 5.2 with randomly generated transforms, using uniform distribution, results in similar graphs. We see this in Figures 5.3 and 5.4.

The theoretical framework suggests to use transforms generated by the positive-frequency coefficients of functions with digital length equal to the digital length of the signals that are to be transformed. Figures 5.1 through 5.4 on the other hand, suggest that the transforms generated by all Fourier coefficients of functions of digital length equal to half of the length of the digital signals might be superior. If we consider transforms generated by normally distributed random numbers, however, the choice of digital length does not appear to be important. In both cases, approximating the Dirac delta function using a partial sum of bands results in significant distortion. We see this in Figures 5.5 and 5.6.

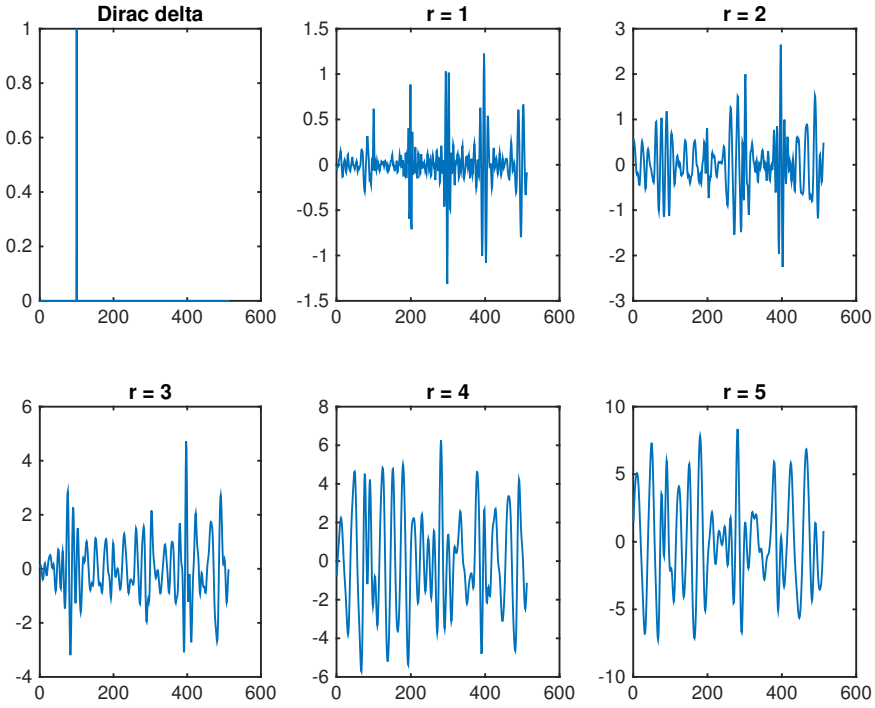

Figure 5.1: Filter Profile of $w_{512} = \sin(4t)e^{-(t-\pi)^2} + 1$.

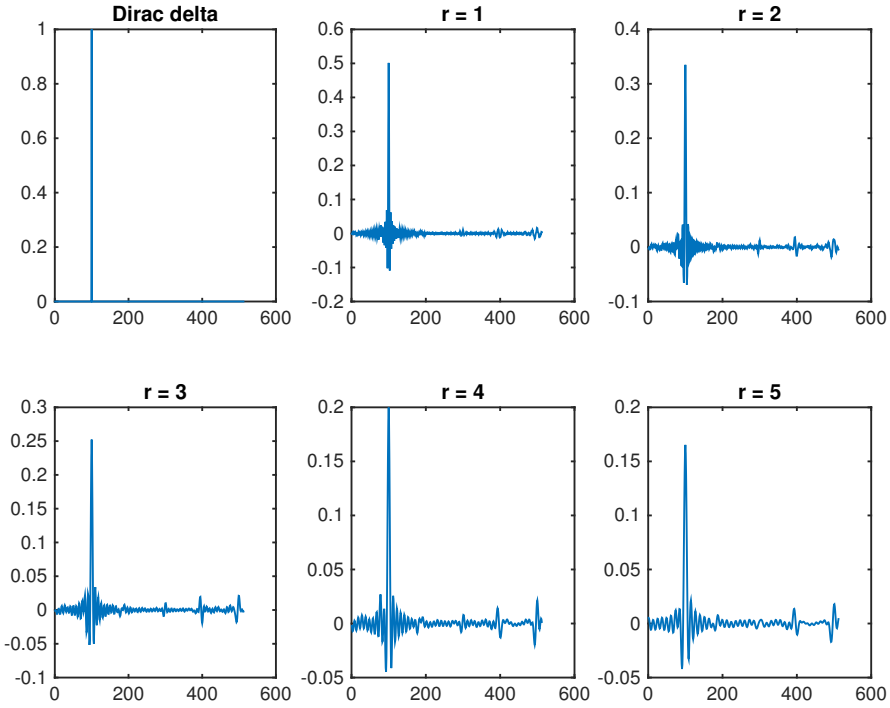

Figure 5.2: Filter Profile of $w_{256} = \sin(4t)e^{-(t-\pi)^2} + 1$.

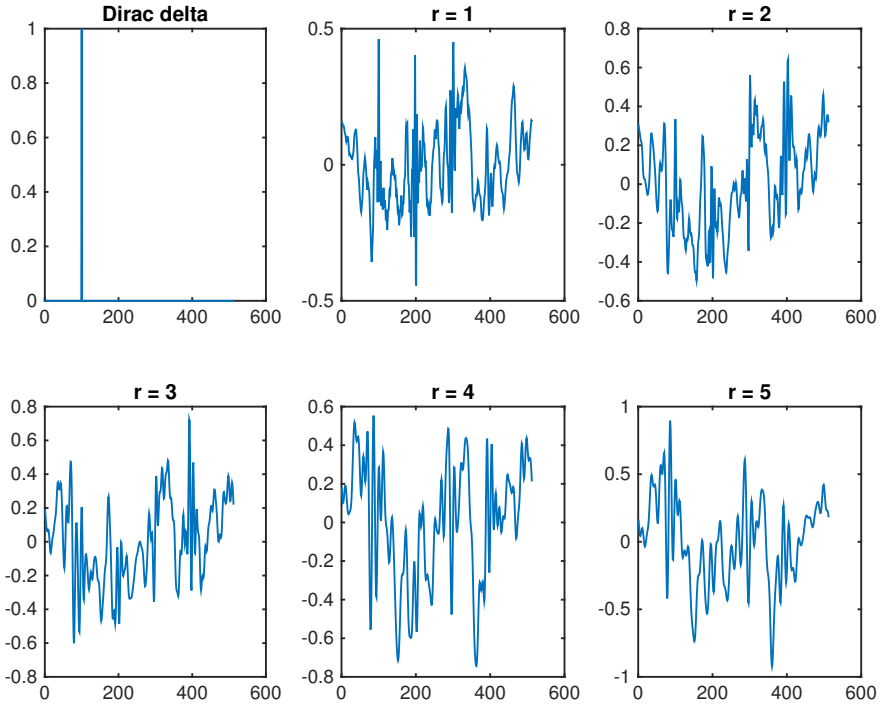

Figure 5.3: Filter Profile of w_{512} = rand(512,1).

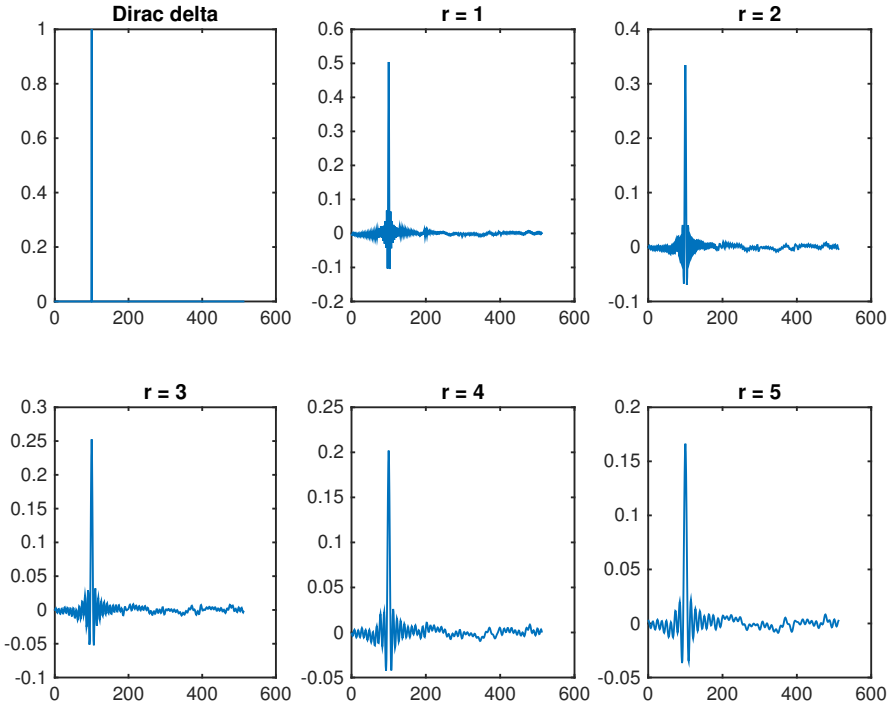

Figure 5.4: Filter Profile of w_{256} = rand(256,1).

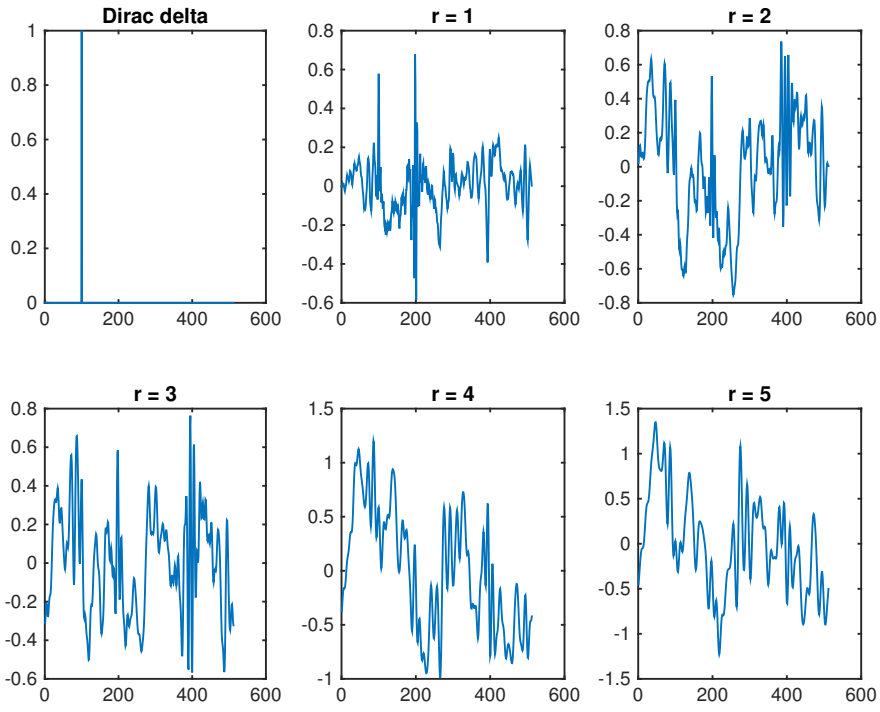

Figure 5.5: Filter Profile of w_{512} = randn(512,1).

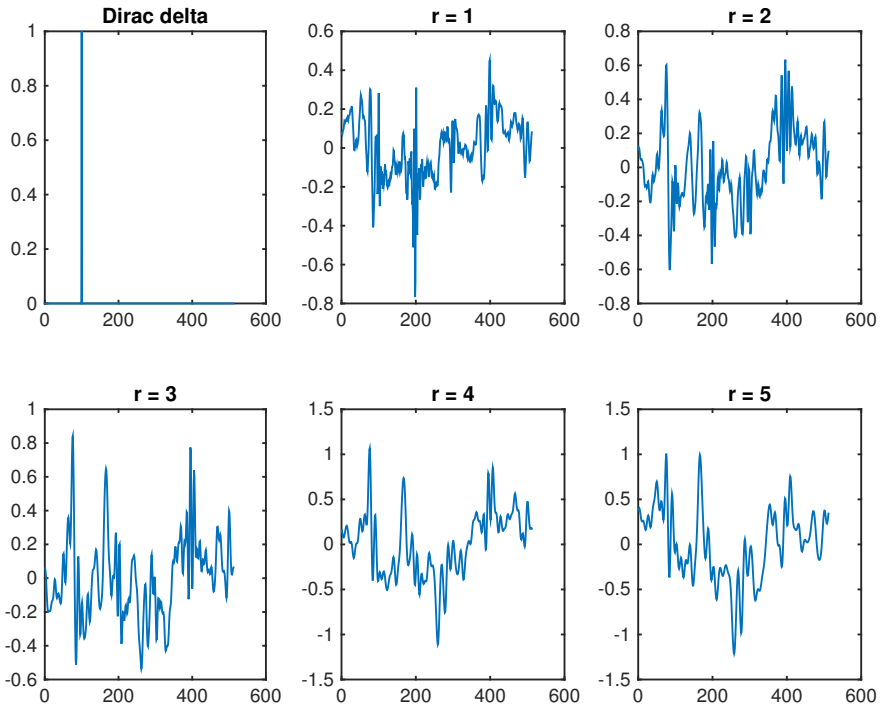

Figure 5.6: Filter Profile of w_{256} = randn(256,1).

The Dirichlet or periodic sinc function

Next we will consider the effects of the P-transform filters on the Dirichlet function, or periodic sinc function, given by

$$D_N(t) = \begin{cases} \frac{sin(Nt/2)}{Nsin(t/2)} & t \neq 2\pi k, k = 0, \pm 1, \pm 2, \pm 3, ... \\ (-1)^{k(N-1)} & t = 2\pi k, k = 0, \pm 1, \pm 2, \pm 3, ... \end{cases} \tag{5.6.3}$$

for any nonzero integer N. This function has period 2π for odd N and period 4π for even N. We will use the same generating waveforms as in the case of the Dirac delta function.

In Figures 5.7 through 5.12 the signal is the Dirichlet function for $N = 6$ over the interval $[0, 2\pi]$, while the P-transform filters are obtained by excluding bands 1 through 5. Note that the Dirichlet function D_6 is not periodic over $[0, 2\pi]$. The results of the filters are similar to those in the case of the Dirac delta function.

In Figures 5.13 through 5.18 the signal is again the Dirichlet function for $N = 6$, but now over the interval $[0, 4\pi]$, where we have a full period of D_6. Interestingly enough, regardless of the choice of generating waveform, we are able to approximate the signal with only few orthogonal band projections. In all cases, the signal is significantly distorted when more than 26 bands are removed. Note that a D-matrix of size 256×256 has 31 orthogonal bands. The exclusion of 26 bands amounts to the exclusion of the last 214 columns of the D-matrix, that is, the retention of the 5 bands which contain the first 42 columns of the matrix.

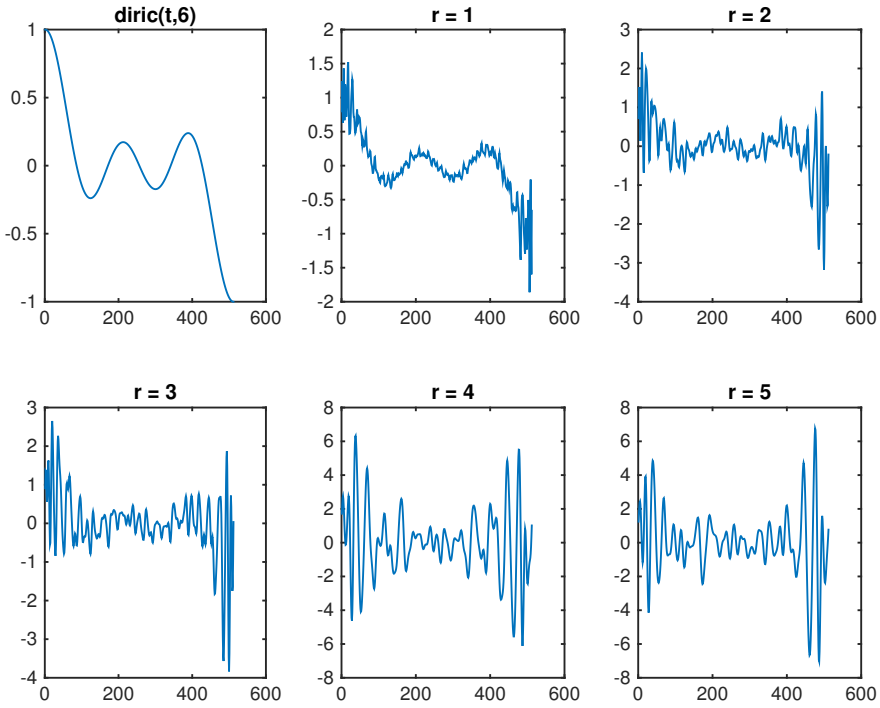

Figure 5.7: Signal D_6 over $[0, 2\pi]$, waveform $w_{512} = \sin(4t)e^{-(t-\pi)^2} + 1$ over $[0, 2\pi]$.

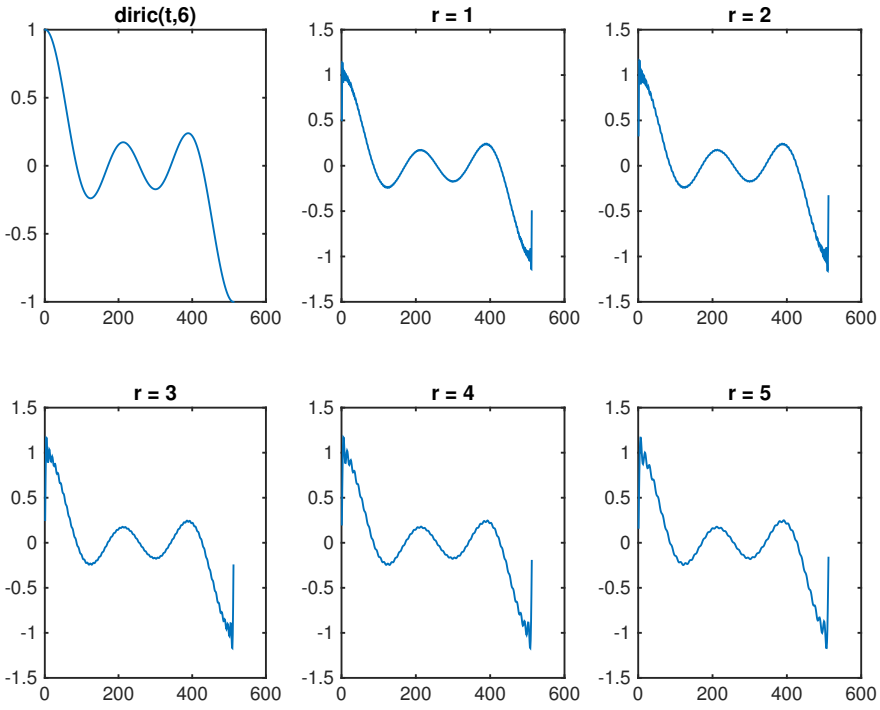

Figure 5.8: Signal D_6 over $[0, 2\pi]$, waveform $w_{256} = \sin(4t)e^{-(t-\pi)^2} + 1$ over $[0, 2\pi]$.

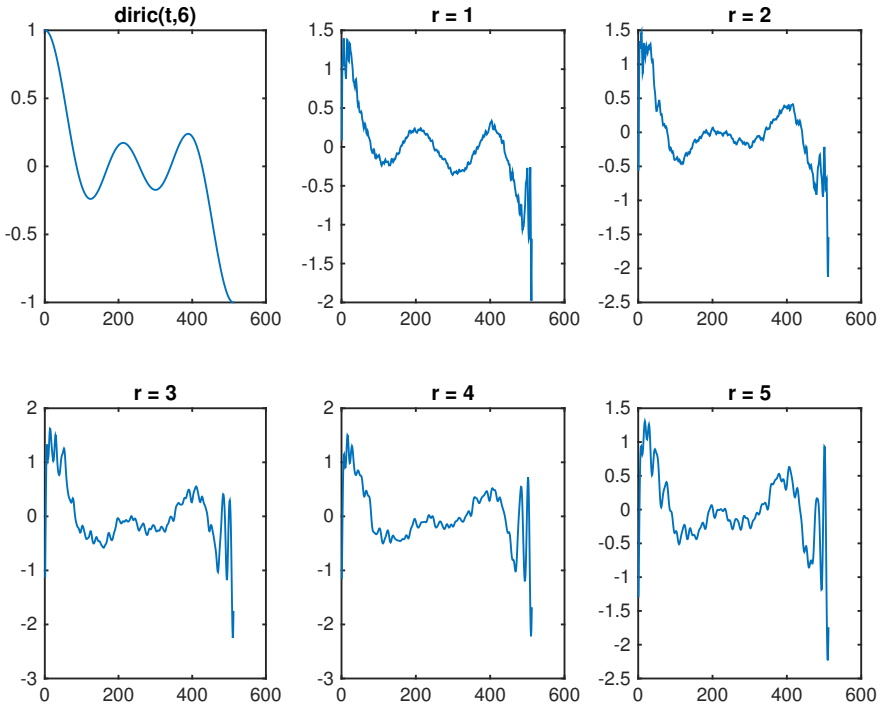

Figure 5.9: Signal D_6 over $[0, 2\pi]$, waveform $w_{512} = \text{rand}(512,1)$.

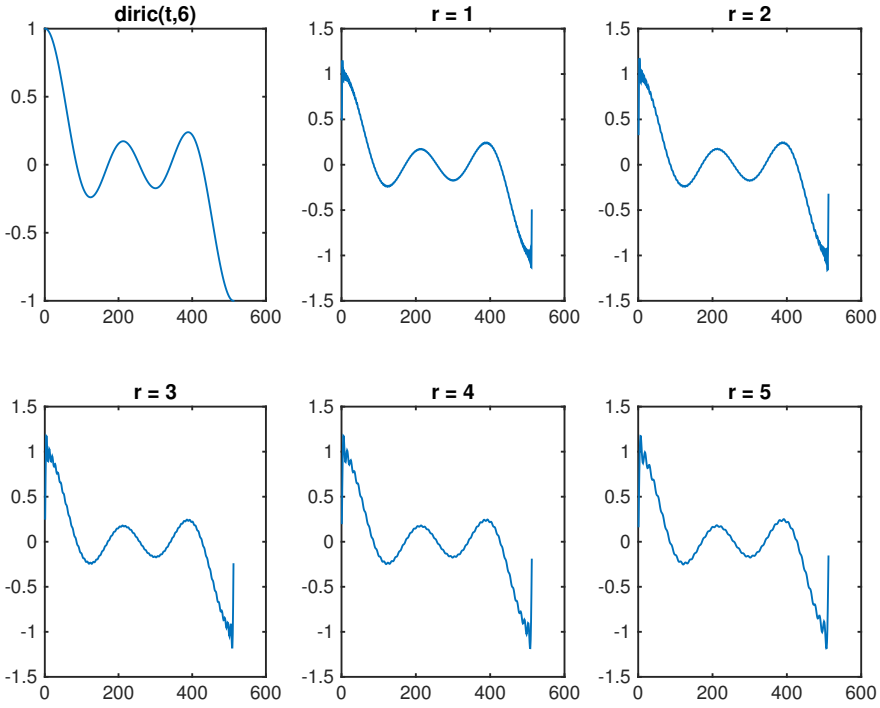

Figure 5.10: Signal D_6 over $[0, 2\pi]$, waveform w_{256} = rand(256,1).

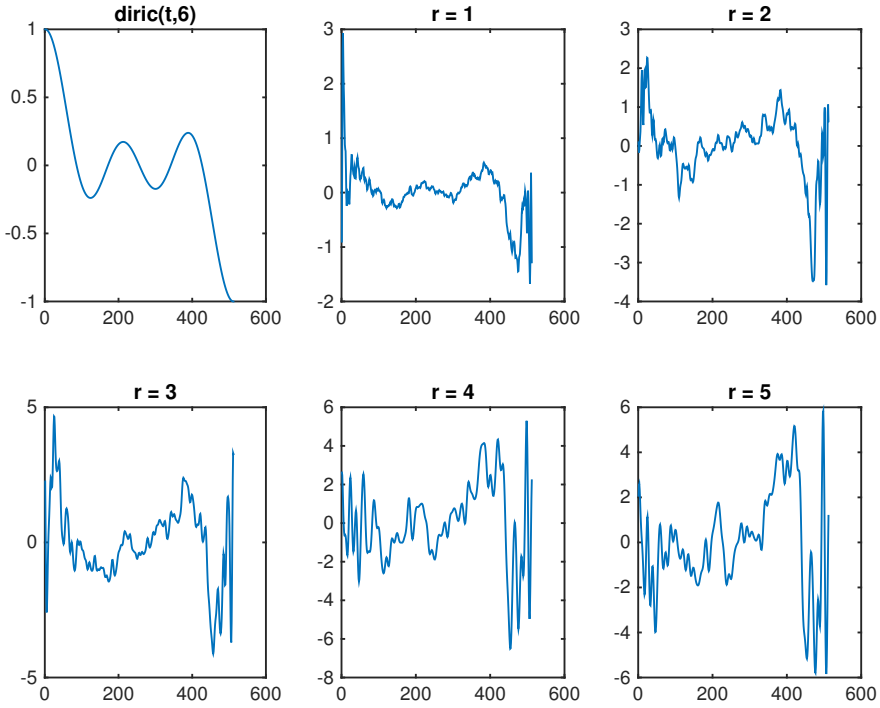

Figure 5.11: Signal D_6 over $[0, 2\pi]$, waveform w_{512} = randn(512,1).

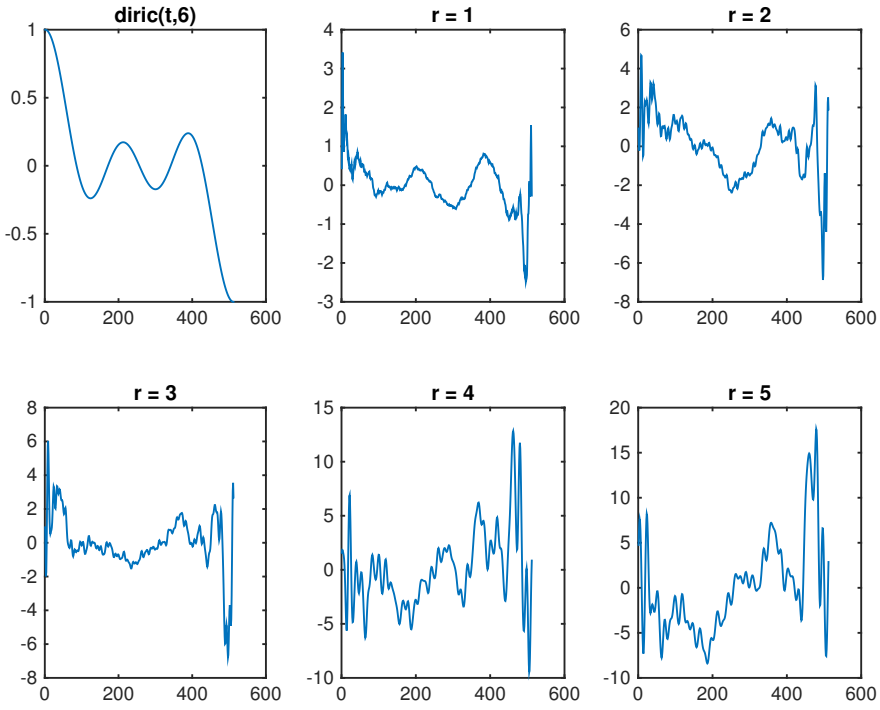

Figure 5.12: Signal D_6 over $[0, 2\pi]$, waveform $w_{256} = \text{randn}(256,1)$.

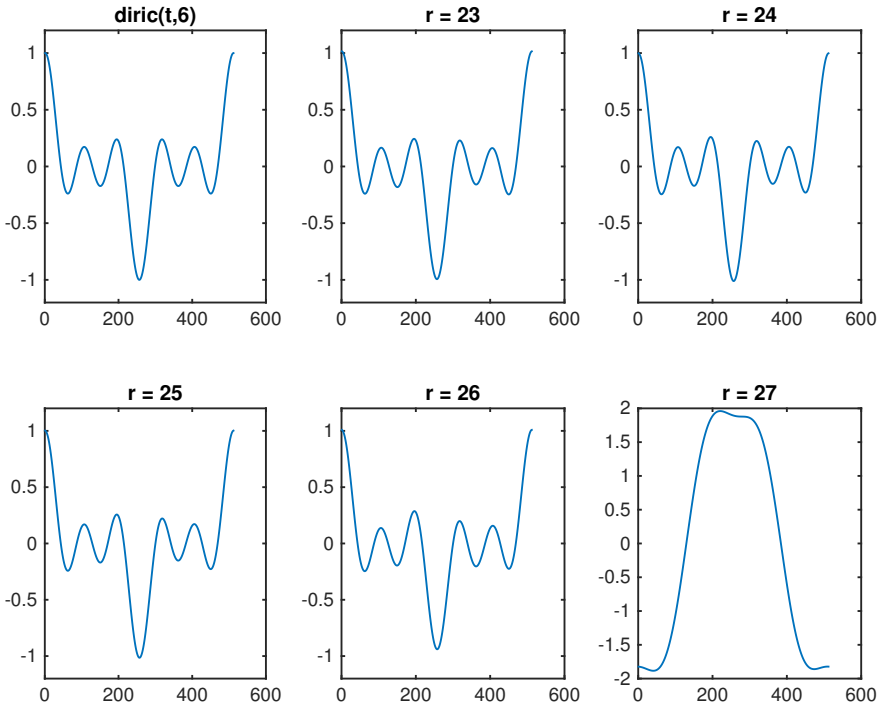

Figure 5.13: Signal D_6 over $[0, 4\pi]$, waveform $w_{512} = \sin(4t)e^{-(t-\pi)^2} + 1$ over $[0, 2\pi]$.

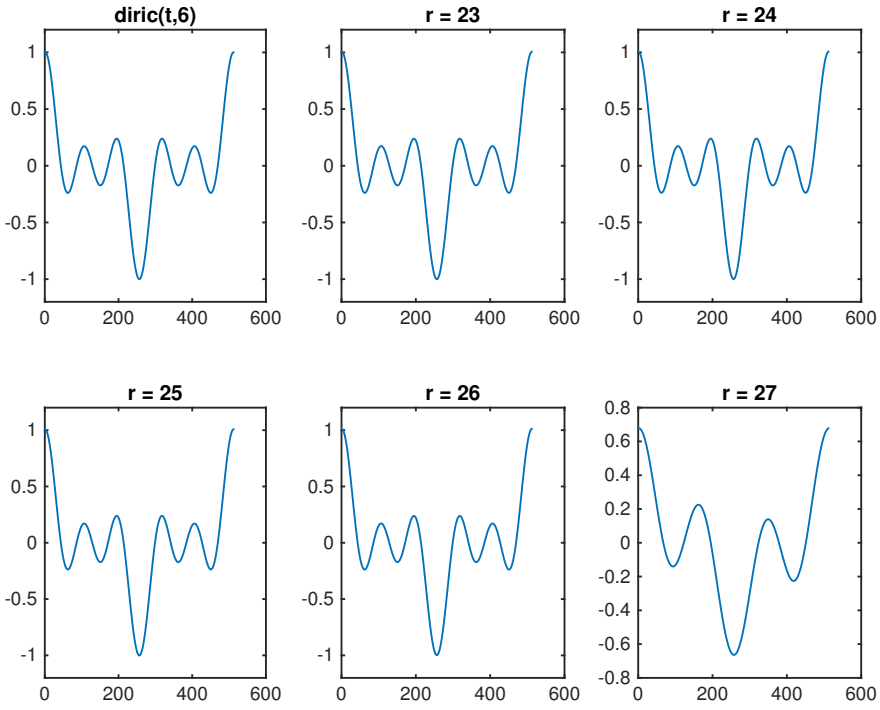

Figure 5.14: Signal D_6 over $[0, 4\pi]$, waveform $w_{256} = \sin(4t)e^{-(t-\pi)^2} + 1$ over $[0, 2\pi]$.

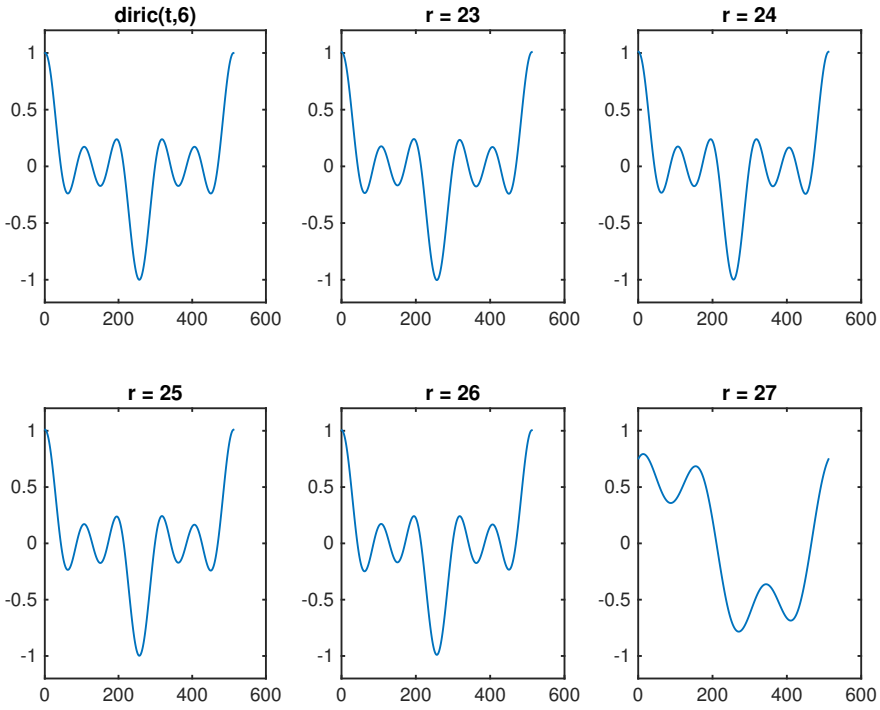

Figure 5.15: Signal D_6 over $[0, 4\pi]$, waveform $w_{256} = \text{rand}(512,1)$.

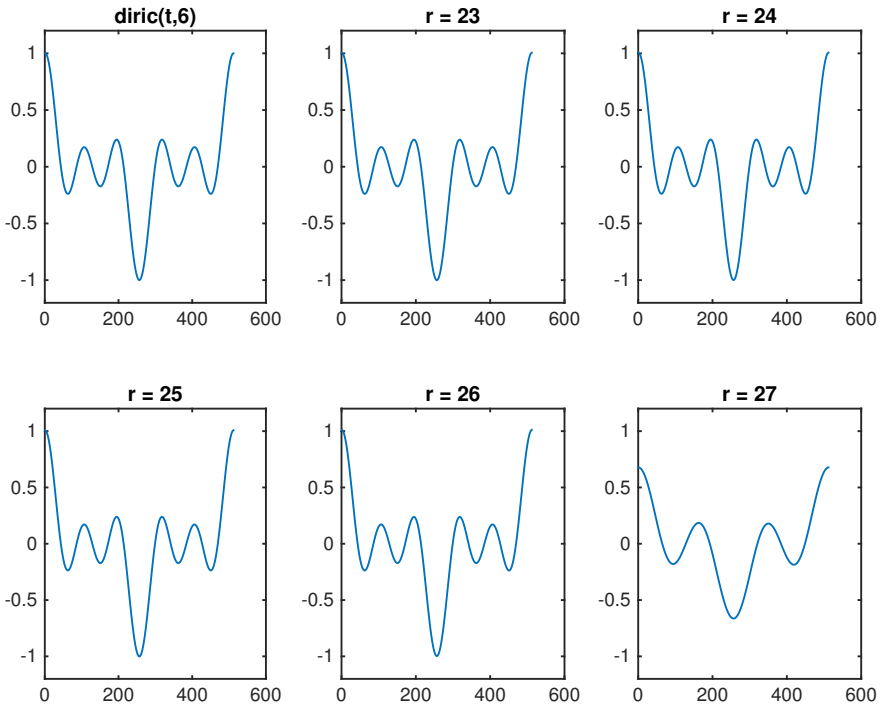

Figure 5.16: Signal D_6 over $[0, 4\pi]$, waveform w_{256} = rand(256,1).

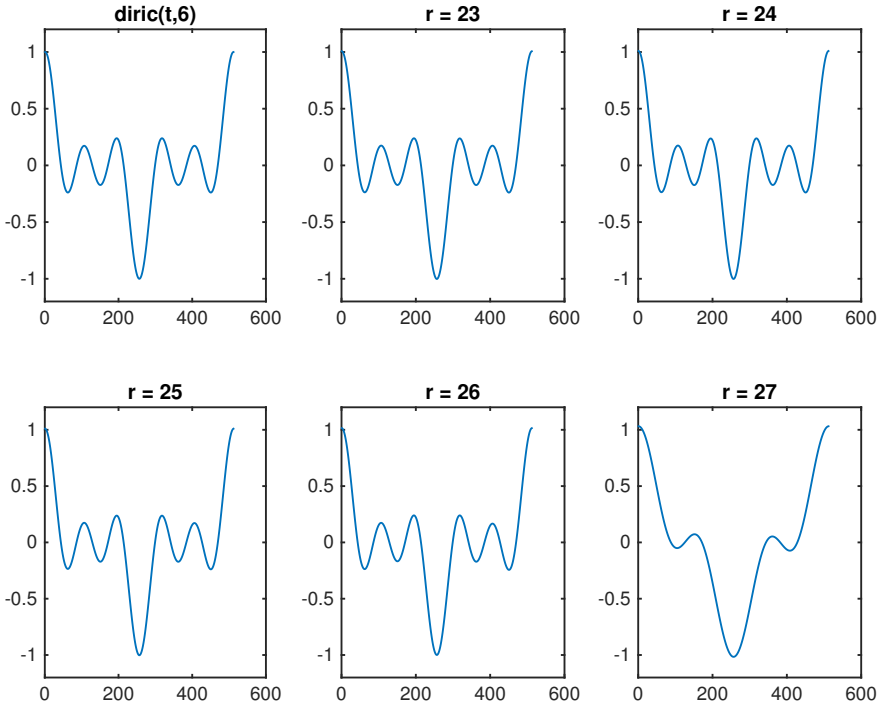

Figure 5.17: Signal D_6 over $[0, 4\pi]$, waveform w_{512} = randn(512,1).

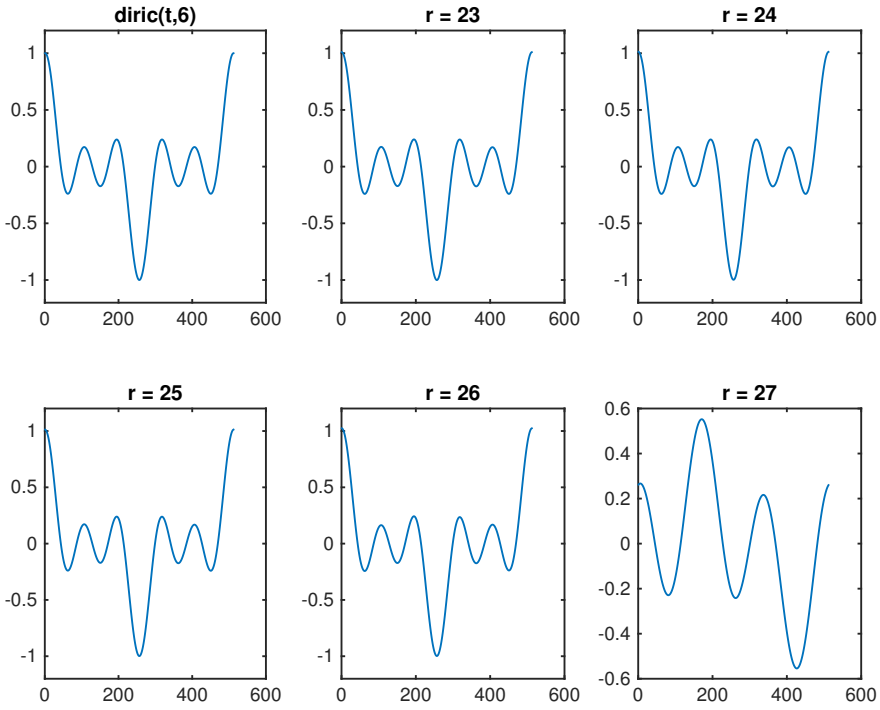

Figure 5.18: Signal D_6 over $[0, 4\pi]$, waveform w_{256} = randn(256,1).

5.6.2 2D Examples

We will close this section with a number of 2D signals transformed using the same P-transform filters as above. To be able to understand the effects of each P-transform filter $P^{-1}P$ for r $= 1, 2, \ldots, 5$ on 2D signals, we will first apply the filters on pure images and then to noisy images.

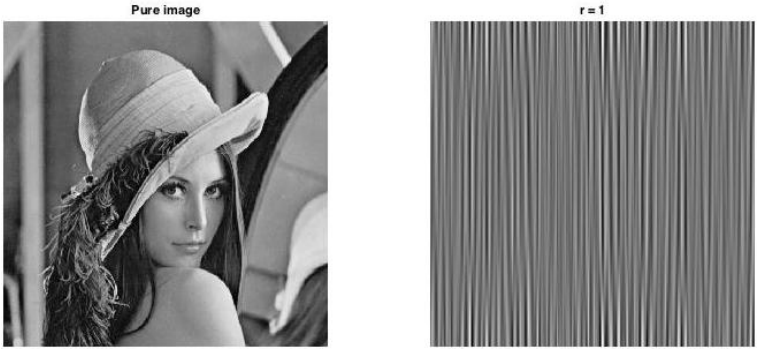

Figure 5.19: 2D Filtering, $w_{512} = \sin(4t)e^{-(t-\pi)^2} + 1$.

In Figure 5.19, the removal of the last band results in a dramatic distortion of the pure image. There is no hope that this filter can be useful in the removal of noise in the case of noisy images.

Figure 5.20: 2D Filtering, $w_{256} = \sin(4t)e^{-(t-\pi)^2} + 1$.

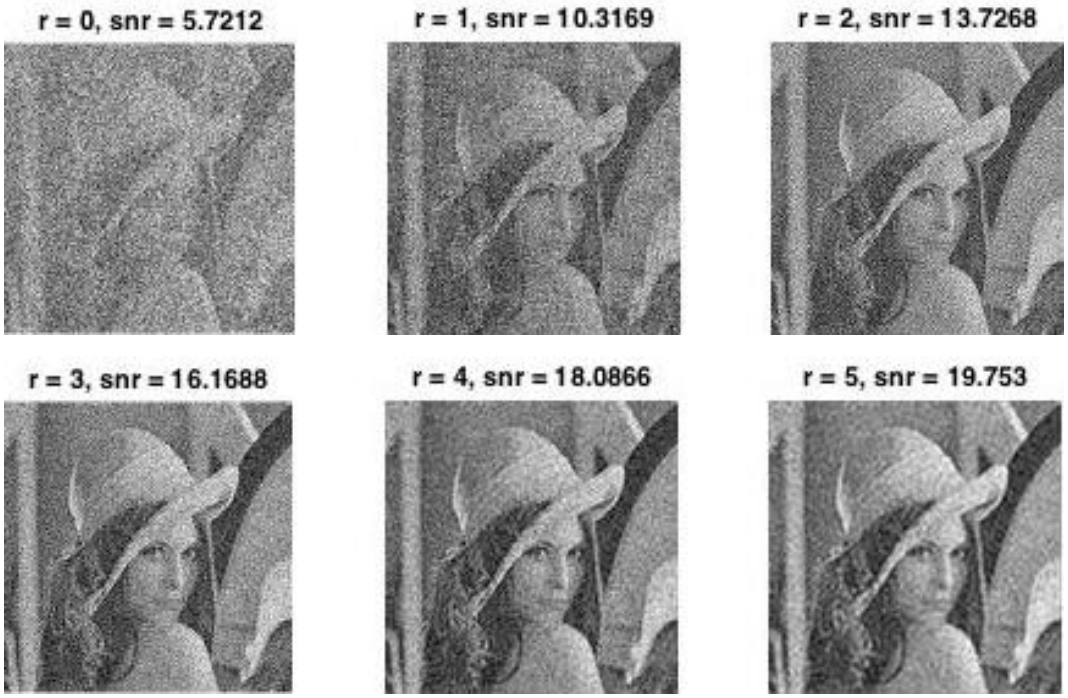

Figure 5.21: 2D Denoising, noise = 80*randn(512), $w_{256} = \sin(4t)e^{-(t-\pi)^2} + 1$.

As in the 1D case, replacing the waveform $\sin(4t)e^{-(t-\pi)^2}+1$ with uniformly distributed random numbers results in simiral outcomes. When we use w_{512} = rand(512,1) to generate the P-transform filters, the pure image is dramatically distorted, while when we use w_{256} = rand(256,1) as the generating waveform, the outputs are drastically improved. Figures 5.22, 5.23 and 5.24 summarize the outcomes of using uniformly distributed random numbers to generate the P-transforms.

Figure 5.22: 2D Filtering, w_{512} = rand(512,1).

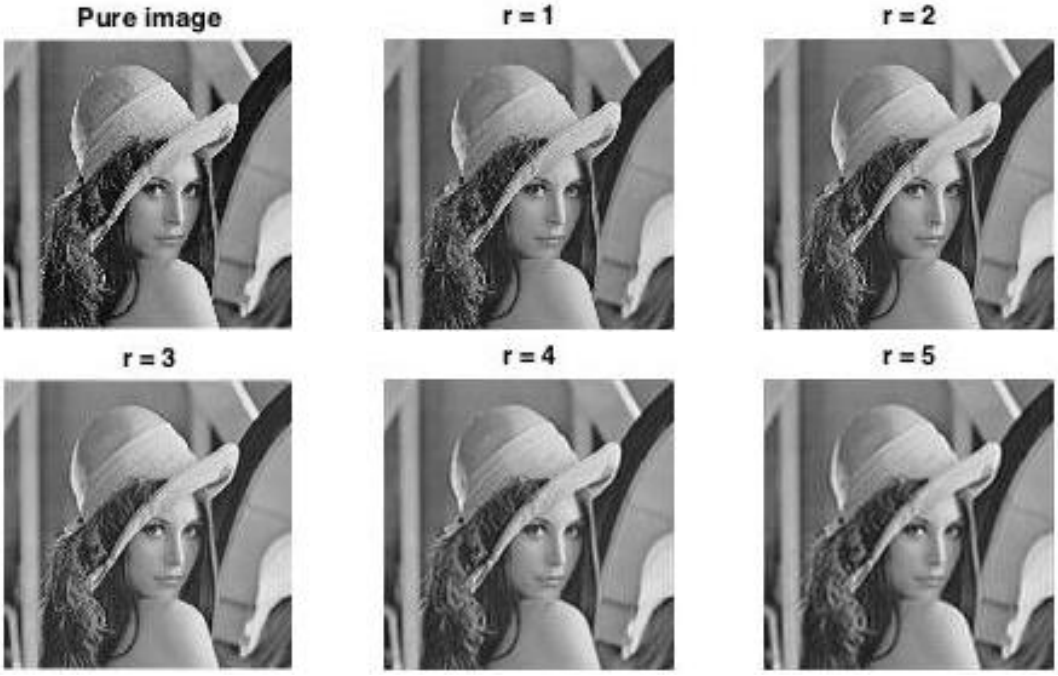

Figure 5.23: 2D Filtering, w_{256} = rand(256,1).

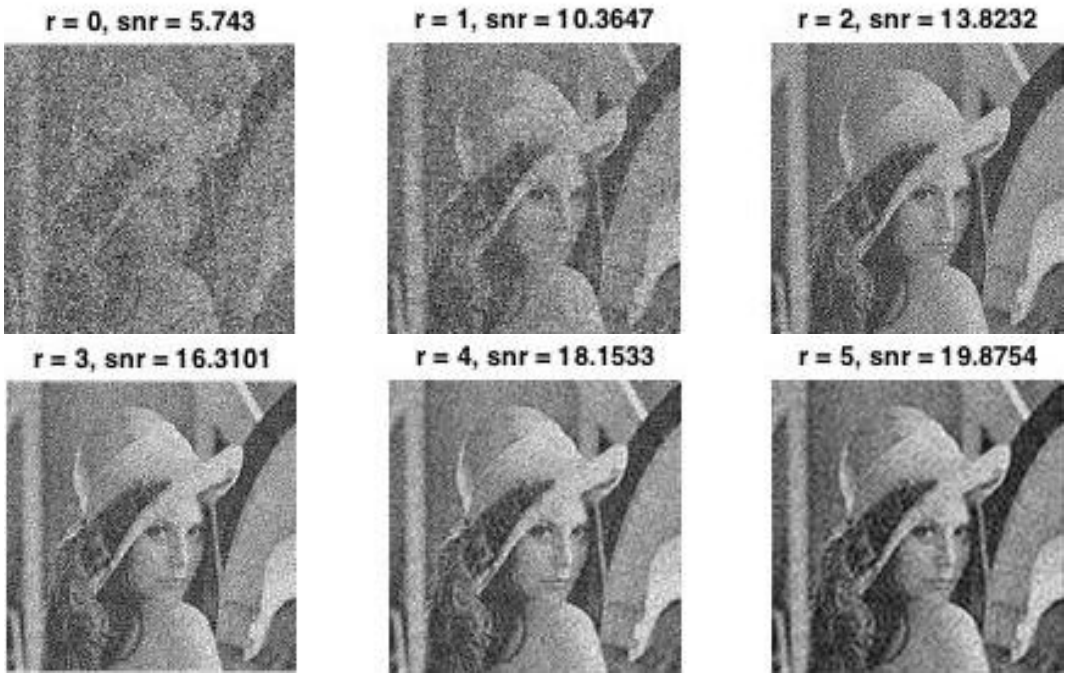

Figure 5.24: 2D Denoising, noise = 80*randn(512), w_{256} = rand(256,1).

Again, as in the 1D case, when we use normally distributed random numbers to generate the P-transforms, there appears to be little difference between the effects of using w_{512} or using w_{256}. Curiously, when we apply the transforms to pure images, w_{512} = randn(512,1) significantly outperforms its uniformly distributed counterpart. It also outperforms w_{256} = randn(256,1). When the signal contains noise, the P-transform filters of either waveform reduce the signal-to-noise ratio, although dramatically more so in the case of w_{256} = randn(256,1).

Figure 5.25: 2D Filtering, w_{512} = randn(512,1).

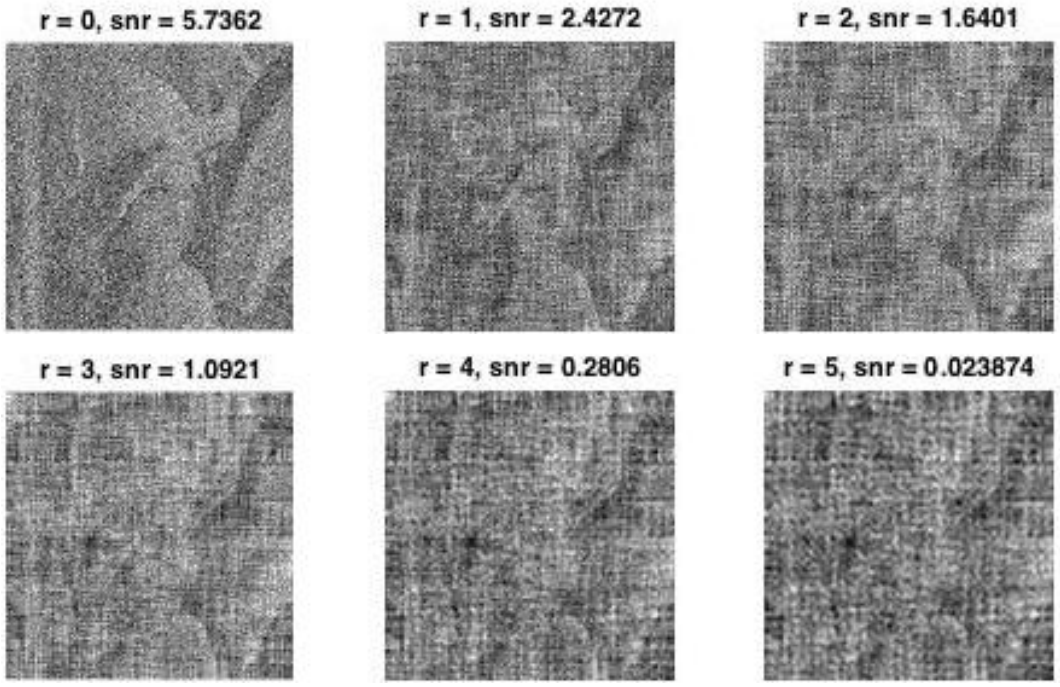

Figure 5.26: 2D Denoising, noise = 80*randn(512), w_{512} = randn(512,1).

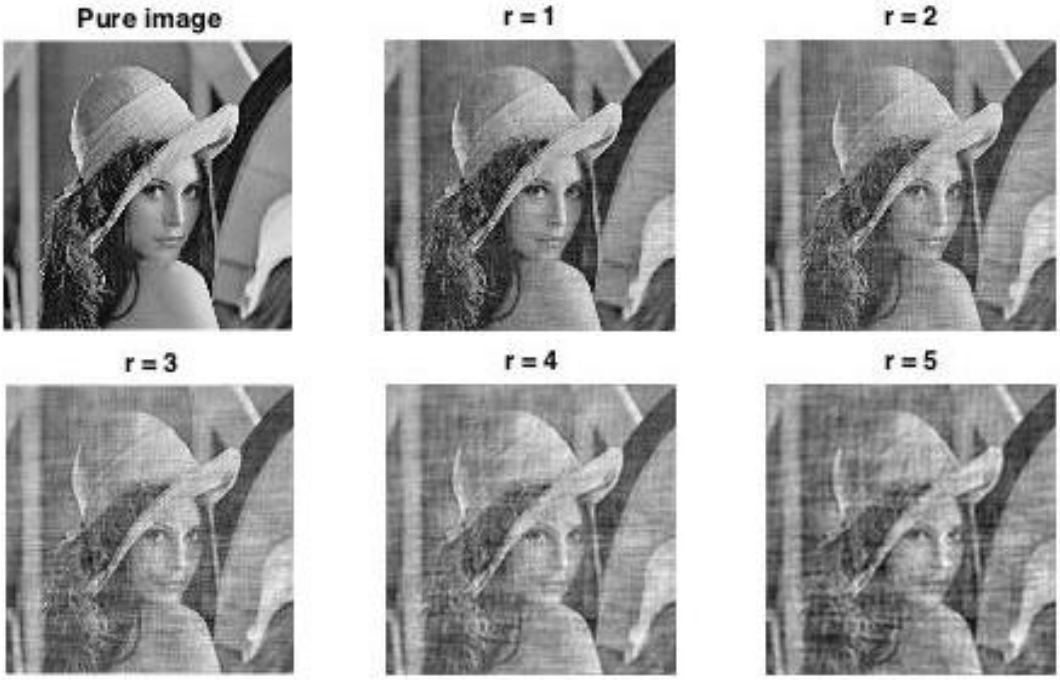

Figure 5.27: 2D Filtering, w_{256} = randn(256,1).

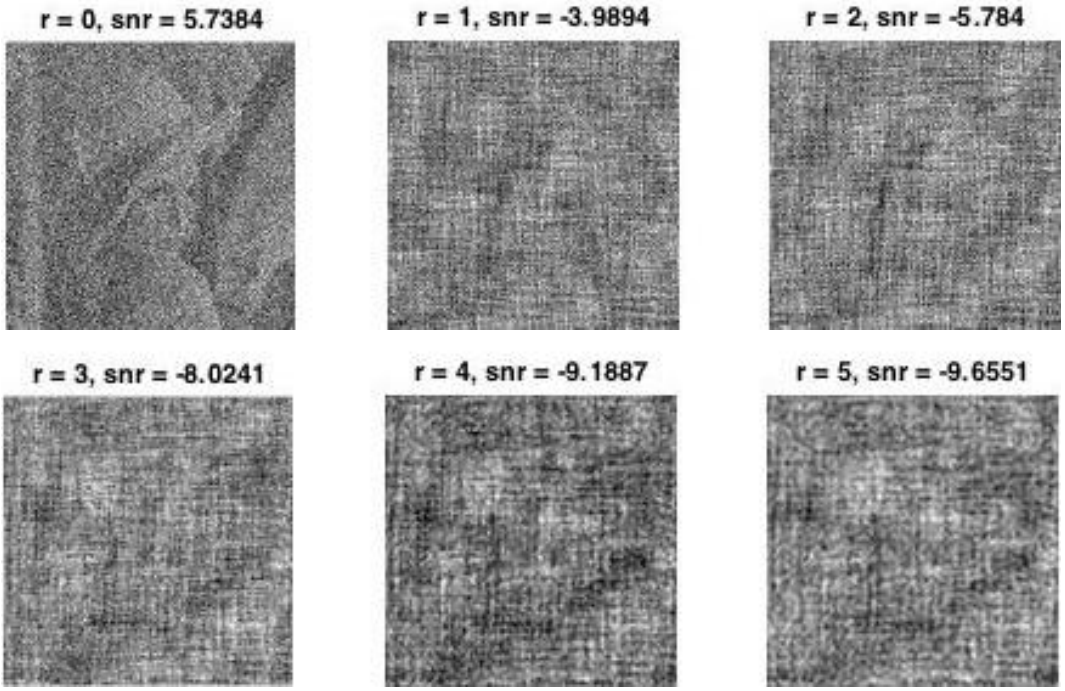

Figure 5.28: 2D Denoising, noise = 80*randn(512), w_{256} = randn(256,1).

Chapter 6

Conclusion

The main objective of my research has been to develop the mathematical tools necessary for the practical application to signal processing of a new family of non-orthogonal transforms, called D-transforms. The potential impact that my work could have on Medical Imaging has been a source of inspiration and motivation from the very beginning. For example, when using conventional X-ray or Computer Tomography imaging, the human body is exposed to radiation which we know to have adverse health effects. Magnetic Resonance Imaging (MRI) is a safe alternative, however, it is often not a viable option. In Canada, for example, the waiting times for an MRI scan range between 12 - 18 months, while the basic scans in private clinics cost $900 on average. In the case of young children, general anesthesia is routinely administered, resulting in additional costs and potential anesthesia related complications. Reducing the processing time of MRI will have a direct consequence in cost reduction and increased accessibility to a safe diagnostic method.

One way in which the processing time of MRI can be shortened is Compressed Sensing or Compressive Sampling, [1], [3], [7], [15]. The idea behind Compressed Sensing, is that some of the data acquired during the MRI scan is redundant and should, therefore, not be collected at all. The task of identifying and sampling only the necessary data relies on the existence of a basis in which the signals admit a sparse representation. The expectation is that we can construct specific D- and P-transforms that will result in a sparse representation for signals generated by MRI. Significantly, the basis representations of real-valued signals, in both the D-bases and P-bases, when generated by a single waveform, exhibit the same conjugate symmetry as the corresponding Fourier coefficients, which can be exploited in Compressed Sensing techniques.

Thus far, I have been able to establish and advance the algebraic and numerical aspects of D- and P-transforms in the finite digital setting. In my subsequent work I intend to go deeper into the analytic structures of these transforms, as well as to take them further towards practical engineering applications. As D-transforms generalize the Fourier transform, it is important to establish the mathematical properties of D- and P-transforms that are similar to the Fourier transform, such as the effects to the scaled, translated and delayed signal, convolution property, correlation property, energy conservation property and Plancherel's theorem, e.g., in the case of D-transforms (and likely also P-transforms) one can expect approximate energy conservation properties and unconditionality of the bases under suitable assumptions, [27]. Further, it is important to establish the correspondence between filtering in physical space, frequency space and the D- and P-transform spaces, including important filtering techniques such as smoothing and sharpening. This

would provide a further introduction of D- and P-transforms from the signal processing perspective, allowing new users to understand these families of fast transforms and be aware of their pros and cons.

Once D- and P-transforms are introduced from the signal processing perspective, I expect that the question of how to design D- and P-transforms for optimal sparse representations of signals will be easier to tackle as well. This, in turn, will advance the study of D- and P-transforms in the context of the Broadband Redundancy phenomenon, [25], [26], whose inherent mathematical structure is applicable in Compressed Sensing techniques for the MRI.

The conjugate symmetry of P-matrices motivates yet another direction of study. Namely, since P-matrices are self-adjoint, they generate a class of unitary operators of the form $U(t) = e^{iPt}$. I am curious to see if it is possible to develop fast algorithms for their implementation, as well as to study their potential for image compression and denoising. Furthermore, this opens the possibility of implementations within the framework of quantum computing. In quantum mechanics, observables are represented by self-adjoint operators, whereas unitary matrices are implementable as quantum algorithms [17], [19].

While I am particularly interested to apply D- and P-transforms towards Compressed Sensing techniques of the MRI, it is important to note that D- and P-transforms can be further studied in the more general context of signal processing. To list few examples, it would be useful to develop Spectral theory based on the D- and P-transforms, apply D- and P-transforms for segmentation, compression and filtering of signals, implement the algorithms for D- and P-transforms for real-time signal processing, and develop windowed D- and P-transforms.

Bibliography

[1] Aldroubi, A., Cabrelli, C., Jaffard, S. and Molter, U. (eds): *New Trends in Applied Harmonic Analysis: Sparse Representations, Compressed Sensing, and Multifractal Analysis*, Springer, Cham, Heidelberg, New York, Dordrecht, London, 2016.

[2] Apostol, T. M.: *Introduction to Analytic Number Theory*, Undergraduate Texts in Mathematics, New York-Heidelberg: Springer-Verlag, 1976.

[3] Carmi, A. Y., Mihaylova, L. S. and Godsill, S. J. (eds): *Comressed Sensing / Sparse Filtering*, Springer-Verlag, Berlin, Heidelberg, 2014

[4] Cashwell, E. D. and Everett, C. J.: *The Ring of Number-Theoretic Functions*, Pacific J. Math. 9, no. 4, 975–985, 1959.

[5] Ceschi, R. and Gautier, J.-L.: *Fourier Analysis*, ISTE - Wiley, 2017.

[6] Cooley, J. W. and Tukey, J. W.: *An algorithm for the machine calculation of complex Fourier series*, Mathematics of Computation, 19(90): 297301, 1965.

[7] Donoho, D. L.: *Compressed Sensing*, IEEE Transactions on Information Theory, Vol. 52, Issue: 4, 2006.

[8] Donoho, D.L.: *Unconditional Bases are optimal bases for data compression and for statistical estimation*, Appl. Computational Harmonic Analysis 1, 100-115, 1993.

[9] Gazi, O.: *Understanding Digital Signal Processing*, Springer, Singapore, 2017.

[10] Green, D. N.: *Generating nonorthogonal bases for signal representations*, Circuits Systems Signal Process. 3, 447475, 1984.

[11] Green, D. N. and Bass, S. C.: *Representing periodic waveforms with nonorthogonal basis functions*, IEEE Trans. Circuits Systems 31, 518584, 1984.

[12] Hardy, G.H.; Riesz, M.: *The general theory of Dirichlet's series*, Cambridge Tracts in Mathematics, 18, Cambridge University Press, 1915.

[13] Hsu, P. H.: *Schaum's Theory and Problems: Signals and Systems*, McGraw-Hill, 1995.

[14] Lathi, B.P.: *Signal Processing & Linear Systems*, Berkeley-Cambridge Press, 1998.

[15] Lustig, M., Donoho, D.L. and Pauly, J.M.: *Sparse MRI: The Application of Compressed Sensing for Rapid MR Imaging*, Magnetic Resonance in Medicine, 58(6): 1182-95, 2007.

[16] McGibney, G., Smith, M.R., Nichols, S. T. and Crawley, A.: *Quantitative evaluation of several partial Fourier reconstruction algorithms used in MRI*, Magnetic resonance in medicine 30.1 pp 51-59, 1993.

[17] Nielsen, M. A., and Chuang, I.L.: *Quantum Computation and Quantum Information*, Cambridge University Press, 2010.

[18] Olson, T.: *Applied Fourier Analysis: From Signal Processing to Medical Imaging*, Birkhauser/Springer, 2017.

[19] Rieffel, E., and Polak, W.: *Quantum Computing: A Gentle Introduction*, The MIT Press, 2014.

[20] Rudin, W.: *Functional Analysis*, McGraw-Hill, New York, 1991.

[21] Seul, M., O'Gorman, L. and Sammon, M. J.: *Practical Algorithms for Image Analysis*, Cambridge University Press, 2000.

[22] Sowa, A.: *A fast-transform basis with hysteretic features*, IEEE Conference Proceedings: Electrical and Computer Engineering (CCECE), 2011 24th Canadian Conference on, 8-11 May 2011, pp.000253-000257, doi: 10.1109/CCECE.2011.6030449.

[23] Sowa, A.: *Factorizing matrices by Dirichlet multiplication*, Linear Algebra and its Applications, Vol. 438, No. 5, 2385-2393, 2013.

[24] Sowa, A.: *On the Dirichlet matrix operators in sequence spaces*, Applicationes Mathematicae (Institute of Mathematics, Polish Academy of Sciences), Applicationes Mathematicae 44, 2, pp. 185-196, 2017.

[25] Sowa, A.: *Qualitative Analysis of Quantum Signals*, CMS Notes, Vol. 49, No. 5, pp 14-15, 2017.

[26] Sowa, A: *Riemann's zeta function and the broadband strudcture of pure harmonics*, The IMA Journal of Applied Mathematics (Oxford Academic), accepted for publication.

[27] Sowa, A.: *The Dirichlet ring and unconditional bases in* $L^2[0, 2\pi]$, Functional Analysis and Its Applications, Vol. 47, No. 3, pp. 227-232, 2013; Translated from Funktsionalnyi Analiz i Ego Prilozheniya, Vol. 47, No.3, pp 75-81, 2013.

[28] Strang, G.: *Linear Algebra and Its Applications*, Harcourth Brace Jovanovich, San Diego, 1988.

[29] Strang, G.: *Signal Processing for Everyone, Computational Mathematics Driven by Industrial Problems*, Springer Lecture Notes in Mathematics 1739 , V. Capasso, H. Engl, and J. Periaux, eds. 2000.

[30] Strang, G.: *The search for a good basis*, Numerical Analysis 1997, D. Griffiths, D. Higham, and A. Watson, eds., Addison Wesley Longman 1997.

[31] Trefethen, L. N. and Bau, D.: *Numerical Linear Algebra*, SIAM, 1997.

[32] Young, R.M.: *An Introduction to Non-Harmonic Fourier Series*, Academic Press, New York, London, Toronto, Sydney, San Francisco, 1980.